Aussagenlogik und Boolesche Algebra in der Unternehmensforschung

von

Gösta-Bernd Ihde

Göttingen 1966

Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41544311_0044

Aussagenlogik und Boolesche Algebra in der Unternehmensforschung

Dissertation
zur Erlangung des
wirtschaftswissenschaftlichen Doktorgrades
der
Wirtschafts- und Sozialwissenschaftlichen Fakultät
der Georg-August-Universität zu Göttingen

vorgelegt

von

Gösta-Bernd Ihde
aus Rostock

Göttingen 1966

Berichterstatter: Professor Dr. Wolfgang Lücke

Mitberichterstatter: Professor Dr. Adolf Angermann

UNIVERSITEITSBIBLIOTHEEK - LEIDEN -

Tag der mündlichen Prüfung: 31. Oktober 1966

I N H A L T

Seite

EINLEITUNG 1

ERSTES KAPITEL

AUSSAGENLOGIK UND BOOLESCHE ALGEBRA 9

11. Begriffe und Abgrenzung 9

12. Kennzeichen und Stellung aussagenlogischer und BOOLEscher Modelle in der Unternehmensforschung 11

ZWEITES KAPITEL

DAS SYSTEM DER AUSSAGENLOGISCHEN PROGRAMMIERUNG 16

21. Aussagenlogische Grundzüge 16

211. Allgemeine Grundlagen 16

212. Das "Rechnen" mit logischen Aussagen 18

22. Die "Aussagenlogische" Programmierung 26

221. Die aussagenlogische Formulierung von Entscheidungsproblemen 26

222. Das Modell von AKERS und FRIEDMAN als aussagenlogisches Programm 32

2221. Die Fragestellung 32

2222. Die Modellbeschreibung 34

2223. Kritik und Erweiterungen 41

DRITTES KAPITEL

DAS SYSTEM DER BOOLESCHEN PROGRAMMIERUNG 52

31. Der Weg zu BOOLEschen Programmen in der Unternehmensforschung 52

32. Die Grundstruktur BOOLEscher Programme 58

33. BOOLEsche und PSEUDO-BOOLEsche Ansätze 61

Seite

34. BOOLEsche Programmierung 63

341. Mathematische Grundlegung 63

342. Das BOOLEsche Programm für ein Selektionsproblem 80

35. BOOLEsche Programme in Verbindung mit der Theorie der Graphen 85

351. Die Graphendarstellung in betriebswirtschaftlichen Modellen 85

352. BOOLEsche Programme für Probleme, welche durch Graphen abgebildet werden können 90

353. BOOLEsche Interpretation eines graphentheoretischen Modells 93

VIERTES KAPITEL

DIE VERFAHREN DER PSEUDO-BOOLESCHEN PROGRAMMIERUNG 98

41. Einführung 98

42. Das Verfahren von FORTET 101

43. IVANESCUs PSEUDO-BOOLEsche Programmierung 105

431. Das Verfahren 105

432. Erläuterung des Algorithmus an einem numerischen Beispiel 111

433. Die Anwendung auf Netzwerk-Fluß-Probleme als Beispiel für die ökonomische Relevanz der PSEUDO-BOOLEschen Programmierung 114

44. Der "Additive Algorithmus" von BALAS 124

441. Einführung 124

442. Darstellung des Algorithmus 125

443. Praktische Erprobung des "Additiven Algorithmus" 145

SCHLUSS 155

Abkürzungen 156

Literaturverzeichnis 157

EINLEITUNG

Die fehlende Berücksichtigung der Methoden und Erkenntnisse der "neuen Logik" in den Sozialwissenschaften veranlaßte 1936 O. MORGENSTERN zu der Feststellung, daß hier "einer der gewaltigsten und eindruckvollsten Fortschritte" des menschlichen Geistes übersehen worden sei, obwohl seine "Ergebnisse für alle sozialwissenschaftlichen Probleme von größter, ja schlechthin von grundlegender Bedeutung sind."[1] An diesem Umstand aber hat sich bis heute - was die unmittelbare Anwendung "logistischer" Prinzipien in den Wirtschafts- und Sozialwissenschaften betrifft - kaum etwas geändert, obgleich insbesondere die Wirtschaftswissenschaften inzwischen einen tiefgreifenden Prozeß der Formalisierung und Kalkülisierung durchgemacht haben[2].

Die Behauptung O. MORGENSTERNs, daß durch die Anwendung der Mathematik und "neuen Logik" kein "artfremdes Element"[3] in die Wirtschafts- und Sozialwissenschaften hineingetragen wird - obwohl damals sehr umstritten - ist heute, zumindest was die Verwendung der Mathematik angeht, grundsätzlich anerkannt: "Es ist nicht so, daß irgendein fundamentaler Grund bestände, weshalb die Mathematik in der Oekonomik nicht be-

---

[1] Morgenstern, O.: Logistik und Sozialwissenschaft, in: ZfN Bd. VII 1936, S. 1. Wiederabdruck in: Logik der Sozialwissenschaften, Hrsg. Topitsch, E., Köln - Berlin 1965, S. 181 ff.

[2] Zur Frage der Symbolisierung, Formalisierung, Quantifizierung und Kalkülisierung besonders
Kosiol, E.: Modellanalyse als Grundlage unternehmerischer Entscheidungen, in: ZfhF 13. Jg. 1961, S. 318 ff.
Kosiol, E.: Betriebswirtschaftslehre und Unternehmensforschung, in: ZfB 34. Jg. 1964, S. 743 ff.
Zimmermann, W.: Modellanalytische Verfahren zur Bestimmung optimaler Fertigungsprogramme, Berlin 1966, S. 35 ff.
ZIMMERMANN gibt in seiner Dissertation eine Zusammenstellung über die wesentlichen Stellungnahmen zur "logisch hinreichend" begründeten Anwendung der Formalwissenschaften im betriebswirtschaftlichen Bereich.

[3] Morgenstern, O.: Logistik und Sozialwissenschaften, a.a.O., S. 3.

nutzt werden könnte."[1]

Der zunehmenden Komplexität der betrieblichen Probleme in einer hochentwickelten arbeitsteiligen Wirtschaft begegnet die theoretische Betriebswirtschaftslehre durch die Anwendung neuer und weiter entwickelter Darstellungsmethoden und Erklärungs- bzw. Lösungsmodelle, indem sie "bestehende Hemmungen der Methodenbeschränkung überwindet und unvoreingenommen und mutig in unerschlossenes Neuland vorstößt."[2]

Die Anwendung und die Erfolge der numerischen Verfahren der Unternehmensforschung sind allgemein anerkannt. Immer mehr betriebswirtschaftliche Probleme werden in eine formalisierte Darstellung übertragen und kalkülartigen Lösungsverfahren zugänglich gemacht: "Die Formalwissenschaften Mathematik, Statistik und Logik wenden sich den neuen Fragestellungen und Kalkülmethoden fortschreitend zu und erarbeiten dafür bisher nicht vorhandene Lösungswege."[3] Mathematik und Statistik haben inzwischen auch ihren festen Platz in der betriebswirtschaftlichen Forschung. Im zunehmenden Maße werden Begriffe und Vorstellungen aus der technischen Nachrich-

---

[1] Neumann, J. v. und Morgenstern, O.: Spieltheorie und wirtschaftliches Verhalten, Würzburg 1961, S. 3.
Zur Mathematisierung in der Betriebswirtschaftslehre s.a.
Angermann, A.: Unternehmerische Entscheidungen und Operations Research, in: Betriebsführung und Operations Research, Frankfurt/M. 1963, S. 11 ff. und
Schneider, E.: Sinn und Grenzen der quantitativen Wirtschaftsforschung, in: ZfgesSt Bd. 108 1952, S. 594 ff.

[2] Kosiol, E.: Modellanalyse als Grundlage ..., a.a.O., S. 319.

[3] Kosiol, E.: Modellanalyse als Grundlage ..., a.a.O., S. 327.

ten- und Informationstheorie[1] und der Kybernetik[2] in die Betriebswirtschaftslehre übernommen. "Diese Bemühungen sind ein Teil der umfassenden Bestrebungen, die sogenannten 'Informations- und Kommunikationstheorien' auf betriebswirtschaftlichem Sektor nutzbringend anzuwenden."[3]

Die Forderung O. MORGENSTERNs jedoch, sich der "neuen Logik" zuzuwenden, hat offenbar kaum Beachtung gefunden: "A

---

[1]Vgl. Adam, A.: Messen und Regeln in der Betriebswirtschaft, Würzburg 1959.
Stahlmann, J.: Organisation, Entscheidung und Kommunikation, Diss. Göttingen 1960.
Kramer, R.: Die betriebswirtschaftliche Bedeutung von Information und Kommunikation, insbesondere für die Struktur des Betriebes, Diss. Mannheim 1962, veröffentlicht auch unter dem Titel: Information und Kommunikation, Berlin 1965.
Adam, A.: Systematische Datenverarbeitung bei der Auswertung von Versuchs- und Beobachtungsergebnissen, Würzburg 1963.
Pietzsch, J.: Die Information in der Unternehmung, Köln und Opladen 1965.
Mertens, P.: Betriebliche Dokumentation und Information, Meisenheim am Glan 1965.
Pierce, J.R.: Phänomene der Kommunikation, Düsseldorf - Wien 1965.

[2]Vgl. z.B. Angermann, A.: Kybernetik und betriebliche Führungslehre, in: BFuP 11. Jg. 1959, S. 257 ff.
Sauter, B.: Die Bedeutung der Kybernetik für die Betriebsorganisation, Diss. Mannheim 1959.
Nürck, R.: Unternehmungsführung - Ein Regelungsproblem, in: BFuP 12. Jg. 1960, S. 230 ff.
Nürck, R.: Steuerung, Regelung, Kybernetik, Wirtschaftskybernetik, in: BFuP 17. Jg. 1965, S. 329 ff.
Nürck, R.: Wirtschaftskybernetik. Ein Schlüssel zur Ganzheitsbetrachtung, in: ZfB 35. Jg. 1965, S. 573 ff.
Pfiffner, M.; Sherwood, F.P.: Administrative Organization, New York 1960.
Adam, A.: Betriebskybernetik, in: Ufo 4. Bd. 1960, S. 44 f.
Beer, S.: Kybernetik und Management, Frankfurt/M. 1963.
Koreimann, D.: Kybernetische Grundlagen der Betriebswirtschaft, in: BFuP 17. Jg. 1965, S. 617 ff.
Stanzky, R.: Kybernetik ökonomischer Reproduktion, Berlin 1966.

[3]Grochla, E.: Zur Organisation des betrieblichen Planungsablaufs, in: ZfB 32. Jg. 1962, S. 714.

further theoretical method, which has yet received little attention," stellt CUSHEN fest[1]. Dies erscheint aus vielen Gründen verwunderlich, bestehen doch zahlreiche Anknüpfungspunkte:

(1) Mit der Übernahme informationstheoretischer und kybernetischer Gedanken in die Betriebswirtschaftslehre werden notwendigerweise "logistische" Ansätze mit übertragen. Kybernetische Systeme nämlich sind "primär durch den Austausch von Informationen gekennzeichnet."[2]

Theoretische Grundlage und mathematische Ausdrucksform informationstheoretischer Aussagen aber ist vor allem die "Logistik"[3]. Bereits der Logistiker NORBERT WIENER, der die wissenschaftlichen Grundlagen der "übergreifenden"[4] Disziplin der Kybernetik erarbeitet hat[5], betont,

---

[1] Cushen, W.E.: Symbolic Logic in Operations Research, in: Operations Research for Management, Hrsg. Closkey, J.F., Trefethen, F.N., Baltimore 1960, S. 187.

[2] Koreimann, D.: Kybernetische Grundlagen ..., a.a.O., S. 621.

[3] Vgl. Taschenbuch der Nachrichtenverarbeitung, Hrsg. Steinbuch, K., Berlin - Göttingen - Heidelberg 1962.
Steinbuch, K.: Automat und Mensch, 2. Aufl., Berlin - Göttingen - Heidelberg 1963, S. 51 ff.
Weyh, U.: Elemente der Schaltungsalgebra, 3. Aufl. München 1964.
Rechenberg, P.: Grundzüge digitaler Rechenautomaten, München und Wien 1964.
Speiser, A.: Digitale Rechenanlagen, 2. Aufl. Berlin - Heidelberg - New York 1965.

[4] Kosiol, E. mit Szyperski, N. und Chmielewicz, K.: Zum Standort der Systemforschung im Rahmen der Wissenschaften, in: ZfbF 17. Jg. 1965, S. 348 u. 374.
Es ist jedoch umstritten, ob die Kybernetik eine selbständige Wissenschaft ist. Vgl. Kramer, R.: Information und Kommunikation, a.a.O., S. 15: "... Kybernetik, die von einigen Autoren - wohl irrtümlicherweise - sogar als selbständige Disziplin bezeichnet wird."

[5] Vgl. Wiener, N.: Cybernetics, New York 1948, Deutsche Übersetzung Düsseldorf 1963.

"daß die Algebra der Logik par excellenz"[1] (d.h. die "neue Logik"), die einfachste Methode ist, um die logischen Möglichkeiten eines kybernetischen Systems abzubilden. Insofern muß sich die theoretische Betriebswirtschaftslehre - bevor kybernetische Gedanken übernommen werden - mit der "neuen Logik" vertraut machen.

(2) Im stärkeren Maße noch wirddie Betriebswirtschaftslehre durch den Einsatz von programmgesteuerten Datenverarbeitungsanlagen mit der "logistischen" Denk- und Darstellungsweise konfrontiert. In diesen Maschinen werden die digitalen Schaltkreise nach den Gesetzen der logischen Funktionen abgebildet. Die Digitalrechner führen alle Operationen nach Maßgabe der eingegebenen Anweisungen entsprechend den "logistischen" Rechenregeln durch. Dabei werden die verschiedenen Operationen auf sukzessive Alternativentscheidungen "logistischer" Art zurückgeführt[2]: "Es zeigte sich, daß sich die BOOLEsche Algebra dazu bestens eignet"[3], denn während das Zählprinzip auf arithmetische Operationen beschränkt ist, erlaubt das logische Prinzip die Bearbeitung aller Aufgaben, die auf "logischen Verknüpfungen beruhen und dazu gehören im weitesten Sinn alle Probleme, für die sich überhaupt eine klare, eindeutige Lösungsanweisung aufstellen läßt."[4]

(3) Auch auf einem ganz anderen Gebiet, dem der quantitativen Terminologie in der Betriebswirtschaftslehre,

---

[1] Wiener , N.: Kybernetik, a.a.O., S. 174.

[2] GEPPERT spricht von Operationen der logischen Entscheidung im Unterschied zu den Einlese- oder Ausgabe- bzw. arithmetischen Operationen. Vgl. Geppert, F.: Allgemeine Grundsätze zur Programmierung von Digital-Rechenautomaten, in: Ufo Bd. 3 1959, S. 213.

[3] Speiser, A.: Digitale Rechenanlagen, a.a.O., S. 10.

[4] Rechenberg, D.: Grundzüge digitaler ..., a.a.O., S. 12.

zeigt sich die Bedeutung der "neuen Logik", denn "bei der Behandlung begrifflicher und terminologischer Probleme dürfte es aber nicht mehr ausreichen, sich nur auf die Aussagen der klassischen Logik - der Begriffslogik im besonderen - zu stützen, ohne die Ergebnisse der neueren Logik ... als Arbeitsgrundlagen heranzuziehen."[1] Als besonders bedeutungsvolle Beiträge der Logistik für terminologische Fragen werden genannt: Bewußte Trennung von Eigenschaften und Relationen und Unterscheidung von Prädikat und Klassenzugehörigkeit[2].

(4) Einen ausdrücklichen Hinweis auf die Bedeutung der Logistik für betriebliche Probleme gibt STAHLMANN, wenn er feststellt, daß "besonders die Organisationstheorie ... unter dem Einfluß der modernen logistischen Methoden entstanden ist."[3]

(5) Zu erwähnen ist auch noch der Versuch, die doppelte Buchhaltung mit Hilfe der logistischen Darstellungsweise als neues mathematisches System zu formulieren[4]. Diese Bemühungen sind jedoch wenig ergiebig.

Die vorliegende Arbeit schließt an keinen dieser Ansatzpunkte an, vielmehr wird versucht, aus der Logistik und der BOOLEschen Algebra neuartige Operations-Research-Verfahren unmittelbar zur Darstellung, Erklärung und Lösung betrieblicher Probleme zu entwickeln, denn "Operations Research can enjoy considerable profit from invoking the methodologies of symbolic logic."[5]

---

[1] Szyperski, N.: Zur Problematik der quantitativen Terminologie in der Betriebswirtschaftslehre, Berlin 1962, S. 15.

[2] Szyperski, N.: Zur Problematik ..., a.a.O., S. 15.

[3] Stahlmann, J.: Organisation ..., a.a.O., S. 105.

[4] Erdmann, A.: Das Gleichungssystem der doppelten Buchhaltung, in: ZfB 30. Jg. 1960, S. 231 ff.

[5] Cushen, W.E.: Symbolic Logic ..., a.a.O., S. 202.

Ein derartiges Vorgehen scheint aus verschiedenen Gründen gerechtfertigt.

Komplexe betriebliche Beziehungs- und Wirkzusammenhänge können auf "logistische" Art eindeutig und ohne Gefahr von Weitschweifigkeiten kalkülartig formuliert werden[1]. Für besondere Probleme kann dieser Ansatz zum Ausgangspunkt für ein systematisches Lösungsverfahren werden. Sofern nur auf die logischen Abhängigkeiten ohne quantitative Zuordnungen abgestellt wird, liegt die Frage nahe, ob mit Hilfe der "logistischen" Technik nichtnumerische Probelme operationsanalytisch gelöst werden können, zumal "... the procedures of the propositional calculus of mathematical logic (and the analogous Boolean algebra) are becoming of fundamental importance in ... industriel fields, where complex logical, non-numerical problems often arise more frequently than do problems of a numerical nature."[2]

Weiter bestehen zwischen einigen bekannten Entscheidungsmodellen der Unternehmensforschung und den "logistischen" Ansätzen grundsätzliche Beziehungen, teilweise ermöglicht die Anwendung oder zusätzliche Beachtung der "neuen Logik" eine Vereinfachung oder Umformung der herkömmlichen Modelle. Insofern ist die "Logistik" "a preface to operations research, not a substitute."[3]

Vor allem aber gibt es - wie wir ausführlich darstellen werden - eine Reihe wichtiger betrieblicher Fragestellungen und Entscheidungsprobleme, deren logische Grundstruktur zweiwertig ist, deren Lösungen also durch eine Abfolge von

---

[1] Vgl. Lindsay, F.A.: New Techniques for Management Decision Making, New York - London - Toronto 1963, S. 59: "Very complex logical problems can be described and analyzed, just as the more familiar algebra provides a symbolic means of describing and solving numerical problems."

[2] Ledley, R.S.: Mathematical Foundations and Computational Methods for a Digital Logic Machine, in: JORSA Vol. 2 1954, S. 24.

[3] Cushen, W.E.: Symbolic Logic ..., a.a.O., S. 202.

Ja-Nein-Alternativentscheidungen gebildet werden[1]. Die "logische Algebra" als mathematisches System, dessen hervorstechendes Merkmal die Zweiwertigkeit ist, eröffnet hier neue Möglichkeiten für adäquate Lösungsverfahren.

Das Bemühen, Entscheidungsmodelle auf der Grundlage dieser neuen Logik zu formulieren, ist gleichzeitig ein Ausdruck für den Einfluß den technische Notwendigkeiten, d.h. hier die Frage der Programmierbarkeit für Datenverarbeitungsanlagen, auf die Arbeit der Mathematiker und Oekonomen auszuüben, unabhängig davon, daß letzten Endes bei geeigneter Codierung ja alle Verfahren für die Rechenanlagen programmiert werden können.

Zur Beantwortung der gestellten Fragen wird zunächst eine knappe, auf die Erfordernisse der Arbeit abgestellte Einführung in die "neue Logik" gegeben. Das ist notwendig, da im Gegensatz zu den numerischen und nicht-BOOLEschen Techniken sowohl die "Verwendung nichtnumerischer Verfahren ... neu ist"[2] als auch die der BOOLEschen Algebra in der Unternehmensforschung. Notwendig ist auch eine Einordnung der "Logistik" in das System der betriebswirtschaftlichen Untersuchungsmethoden und insbesondere eine Analyse der betrieblichen Entscheidungssituationen im Hinblick darauf, inwieweit ihre Struktur die Anwendung der neuen Logik bzw. BOOLEschen Algebra zuläßt. Diese Möglichkeiten sind theoretisch darzustellen und praktisch zu erläutern.

---

[1] Eine typische Ja- oder Nein-Entscheidung ist z.B. das Urteil über die Liquiditätslage einer Unternehmung, worauf LÜCKE ausdrücklich hinweist und was nicht immer klar erkannt wird. Vgl. Lücke, W.: Die Liquidität im Entscheidungsmodell, in: Gegenwartsfragen der Unternehmensführung. Festschrift zum 65. Geburtstag von W. Hasenack, Herne - Berlin 1966, S. 323.

[2] Schweitzer, M.: Probleme der Ablauforganisation in Unternehmungen, Berlin 1964, S. 75.

ERSTES KAPITEL

AUSSAGENLOGIK UND BOOLESCHE ALGEBRA

11. Begriffe und Abgrenzung

Bislang haben wir von "Logistik", "neuer Logik", "BOOLEscher Algebra" und "logischer Algebra" gesprochen und diese Bezeichnungen teilweise synonym verwandt.

Logistik ist die "Ausdehnung der formalen Methoden der Mathematik auf das Gebiet der Logik"[1] oder die "moderne Gestalt der Logik"[2]. Neben den Bezeichnungen mathematische und moderne Logik wird - entsprechend ihren besonderen Kennzeichen - auch von formaler[3] oder symbolischer[4] und theoretischer Logik gesprochen. Von BOOLEschen Algebren spricht man seit GEORGE BOOLE (1815-1864)[5] bei der algebraischen Darstellung und Behandlung logischer Zusammenhänge. Die Bezeichnung Logistik geht auf einen Vorschlag von L. COUTURAT[6] zurück.

---

[1] Hilbert, D. und Ackermann, W.: Grundzüge der theoretischen Logik, 4. Aufl. Berlin - Göttingen - Heidelberg 1959, Einleitung.

[2] Menne, A.: Was ist und was kann Logistik, Paderborn 1957, S. 4.

[3] Vgl. Lorenzen, P.: Formale Logik, 2. Aufl., Berlin 1962, Bochenski, I.M.: Formale Logik, 2. Aufl., Freiburg - München 1962.
Klaus, G.: Moderne Logik. Abriß der formalen Logik, Berlin 1965.

[4] Vgl. Carnap, R.: Symbolische Logik, 2. Aufl., Wien 1960 und vor allem im anglo-amerikanischen Sprachraum "symbolic logic". Maßgebliche Zeitschrift ist das Journal of Symbolic Logic.

[5] Vgl. Boole, G.: The Mathematical Analysis of Logic, Cambridge 1857 und Boole, G.: An Investigation of the Laws of Thought, London 1854.

[6] Couturat, L.: La logique de Leibniz d'apres des documents inedits, Paris 1901

Von FREYTAG-Löringhoff ordnet die neue Logik allein der Mathematik zu und möchte sie darum "Mathematistik"[1] nennen. Aus dem technischen Bereich stammt der Name "Schaltungsalgebra"[2]. Die Bezeichnung Logistik ist inzwischen unzweckmäßig geworden, da hierunter allgemein die Lehre vom Beschaffungs- und Nachschubwesen der Streitkräfte[3] und spezielle Materialbeschaffungs- und -flußprobleme in Betrieben[4] verstanden werden. Im Rahmen dieser Arbeit unterscheiden wir aussagenlogische und BOOLEsche Ansätze und sprechen von logischer oder BOOLEscher Algebra bzw. algebraischer Logik.

Die eigentliche Entwicklung der algebraischen Logik setzt man mit der entscheidenden von LEIBNIZ eingeleiteten Wendung von der klassischen Logik zur mathematischen Behandlung dieser Probleme an[5]. Außer BOOLE sind in diesem Zu-

---

[1] Von Freytag-Löringhoff: Logik. Ihr System und ihr Verhältnis zur Logistik, 3. Aufl., Stuttgart 1961, S. 197.

[2] Vgl. Weyh, U.: Elemente der Schaltungsalgebra, München 1964. Rechenberg, P.: Grundzüge digitaler Rechenautomaten, München - Wien 1964.

[3] Vgl. hierzu besonders den Versuch von O. MORGENSTERN, eine Theorie der Logistik in diesem Sinne zu entwickeln. Morgenstern, O.: Note on the Formulation of the Theory of Logistics, in: Naval Research Logistics Quarterly Vol. 2 1955, S. 129 ff.
Eccles, H.E.: Logistics - What is it? in: NRLQ Vol. 1 1954, S. 5 ff.
Mirasol, N.M.: A Queuing Approach to Logistics Systems, in: JORSA Vol. 12 No. 5 1964, S. 707: "Logistics, as used by the military, deals with the functions and activities associatet with the procedurement and operational support of equipment systems".
Geisler, M.A.: Logistics research and management science, in: MS Vol. 6 1960, S. 444 ff. Als Fachliteratur ist die von der amerikanischen Marine veröffentlichte Zeitschrift: Naval Research Logistics Quarterly zu nennen. Sprachlicher Ursprung ist Logis-Quartier.

[4] Vgl. z.B. Eaton, Paul T.: Logistik im Betrieb. Materialfluß gewinnt an Bedeutung, in: Der Volkswirt 20. Jg. Nr. 24 1966, S. 1028 f.

[5] Zur Geschichte der BOOLEschen Algebra s. besonders Bochenski, J.M.: Formale Logik, a.a.O., S. 312 ff. und Klaus, G.: Moderne Logik, a.a.O., S. 21 ff.

sammenhang zu nennen: AUGUSTUS DE MORGAN (1806 - 1878), GOTTLOB FREGE (1848 - 1929), ALFRED N. WHITEHEAD (1861 - 1941) und BERTRAND RUSSEL (geb. 1872). Mit JOHN ST. MILL (1806 - 1873) und STANLEY JEVONS (1835 - 1882) haben sich auch zwei namhafte Wirtschaftstheoretiker um die Begründung der algebraischen Logik verdient gemacht. Von Anfang an war die lebhafte Entwicklung der algebraischen Logik als logische Untersuchung der Mathematik und mathematische Untersuchung der Logik[1] mit einer kritischen "Überprüfung der Grundlagen der Mathematik verbunden."[2] In neuester Zeit gehen die entscheidenden Impulse zur Weiterentwicklung der BOOLEschen Algebra vor allem von ihrer praktischen Anwendung bei der Programmierung der Rechenanlagen aus.

Erwähnt werden müssen auch noch die interessanten Anwendungen der algebraischen Logik in der Biologie, Rechtswissenschaft, Psychologie und Volkswirtschaftslehre[3].

12. Kennzeichen und Stellung aussagenlogischer und BOOLEscher Modelle in der Unternehmensforschung

Über das eigentliche Wesen und noch mehr über die Bedeutung der algebraischen Logik bestanden und bestehen sehr unterschiedliche Meinungen. Das gilt besonders für das Verhältnis zur klassischen Logik und zur Mathematik und für die

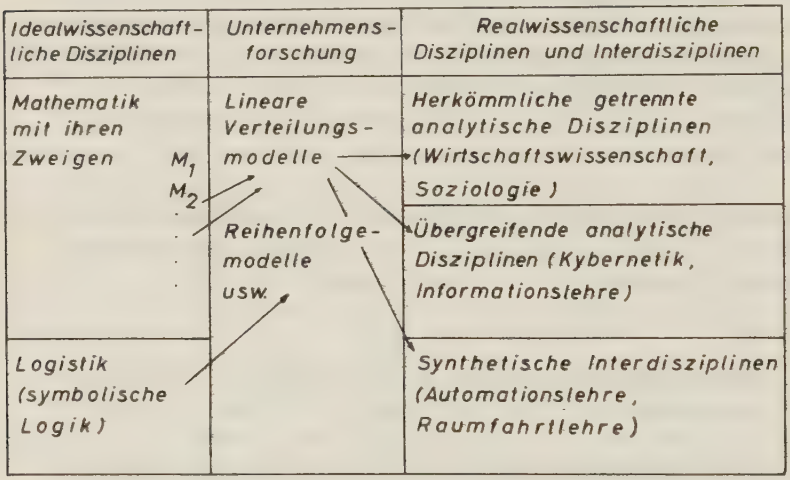
---

[1] Vgl. Courant, Richard und Robbins, Herbert: Was ist Mathematik, Berlin - Göttingen - Heidelberg 1962, S. 90.

[2] Kade, Gerhard: Die logischen Grundlagen der mathematischen Wirtschaftstheorie als Methodenproblem der theoretischen Ökonomie, Diss. Berlin 1958, S. 2.

[3] Vgl. hierzu Bochenski, J.M.; Menne, A.: Grundriß der Logistik, a.a.O., S. 14 und das ausführliche Literaturverzeichnis.

Frage, ob es sich bei der algebraischen Logik um eine selbständige idealwissenschaftliche Disziplin handelt. Diese Meinung vertritt KOSIOL, wenn er sagt: "Die Unternehmensforschung übernimmt aus den verschiedenen idealwissenschaftlichen Disziplinen (Mathematik, Logik) und ihren Zweigdisziplinen geeignete Kalküle"[1] und von mathematischen, statistischen und logistischen Kalkülmodellen spricht. Es ergibt sich dann folgender Zusammenhang[2]:



Figur 1

NEIDHARDT spricht ebenfalls von einer "selbständigen mathematischen Disziplin"[3]. KLAUS sieht in der formalen Logik "nicht nur eine mathematische, sondern auch eine philosophische Disziplin"[4], offenbar jedoch weniger aus wis-

---

[1] Kosiol, E.; Szyperski, N. und Chmielewicz, K.: Zum Standort ..., a.a.O., S. 374.

[2] entnommen aus Kosiol, E.; Szyperski, N. und Chmielewicz, K.: Zum Standort ..., a.a.O., S. 374.

[3] Neidhardt, P.: Informationstheorie und automatische Informationsverarbeitung, 2. Aufl. Stuttgart 1964, S. 183.

[4] Klaus, Georg: Moderne Logik ..., a.a.O., S. 1.

senschaftlicher als aus ideologischer Überzeugung. Nach MENNE wird heute zwischen Logik und logischer Algebra oft nur deswegen unterschieden, weil einige Anhänger der klassichen Logik die von den Mathematikern konzipierte "Logistik" ignorieren[1]. Teilweise wird sogar bezweifelt, daß die Mathematisierung die Logik erweitert hat, was jedoch zweifellos der Fall ist, da bestimmte Gesetze der logischen Algebra in der traditionellen Logik nicht enthalten sind. Die "Logistik" wird dann nur als angewandte Logik und "ihre spezifischen Leistungen mittelbar als solche der Logik"[2] angesehen. Diese Auffassung verkennt offenbar die Ziele der BOOLEschen Algebra; die Anwendung in der mathematischen Grundlagenforschung ist nur ein Teilgebiet.

Nach FISCHL ist die Logik kein Teil der Mathematik, sondern diese nur ein Teil der Logik[3], für BOCHENSKI und MENNE aber gibt es "jedenfalls in der Logistik selbst nichts spezifisch Mathematisches"[4], während sie nach VON FREYTAG-LÖRINGHOFF "unbestreitbar"[5] ein Stück Mathematik ist.

Diese vielfach ins Philosophische hineinspielende Polemik beruht "fast ausschließlich auf Mißverständnissen und mangelnder Sachkenntnis"[6]. Sie führt zu teilweise grotesken Aussagen, wenn z.B. behauptet wird, daß logische Algebra nur von Fachmathematikern betrieben werden könne, die "starke mathematische Begabung, vorzügliche Beherr-

---

[1] Vgl. Menne, A.: Was ist ..., a.a.O., S. 4.

[2] Von Freytag-Löringhoff: Logik ..., a.a.O., S. 197.

[3] Vgl. Fischl, Johann: Logik. Ein Lehrbuch. Mit einem kurzen Abriß über Logistik. 2. Aufl., Graz - Wien - Altötting 1952, S. 140.

[4] Bochenski, J.M.; Menne, A.: Grundriß ..., a.a.O., S. 14.

[5] Von Freytag-Löringhoff: Logik ..., a.a.O., S. 155.

[6] Bochenski, J.M.; Menne, A.: Grundriß ..., a.a.O., S. 12.

schung der logistischen Technik und ausgiebiges Training"[1] mitbrächten, während sie andererseits doch "tatsächlich ... zu den am leichtesten erlernbaren Gebieten der Mathematik"[2] zählt. Mathematisch am logistischen Modell ist zweifellos aber das Bestreben "the over-whelming complexity of some types of management problems"[3] mit Hilfe einer geeigneten Symbolsprache so zu reduzieren, daß innerhalb der bestimmbaren Logikkalküle mittels systematischer Operationsregeln Schlußfolgerungen gezogen werden können[4]. Das besondere Kennzeichen sowohl der aussagenlogischen als auch der BOOLEschen Modelle ist die Tatsache, daß die angewandten Begriffe und Rechenregeln ausschließlich oder überwiegend auf die Aussagenlogik zurückgehen. Die Aussagenlogik ist deshalb der Ausgangspunkt der Untersuchung. Notwendig und zweckmäßig - bislang jedoch nicht durchgeführt - ist hierbei insbesondere eine klare Trennung von aussagenlogischen und BOOLEschen, d.h. algebraischen Ansätzen. Werden allein die logischen Abhängigkeiten berücksichtigt, spricht man von aussagenlogischen Modellen, von BOOLEschen Programmen dagegen, wenn der spezielle Formalismus der BOOLEschen Algebra , also die Mathematik besonderer Art, das Kennzeichen der Ansätze ist. Wenn KOSIOL von logistischen Modellen spricht[5], so sind damit nur die aussagenlogischen gemeint. Diese Auffassung erscheint zu eng. Von größerer Bedeutung für die Unternehmensforschung sind die algebraischen Möglichkeiten der Logik, d.h. die BOOLEsche Algebren.

Sofern von aussagenlogischen Programmen oder Modellen gesprochen wird, kann der Unterschied zu den mathematischen Ansätzen also darin gesehen werden, "daß die Mathematik sich

---

[1] Fischl, Johann: Logik ..., a.a.O., S. 138.

[2] Von Freytag-Löringhoff: Logik ..., a.a.O., S. 155.

[3] Lindsay, F.A.: New Techniques ..., a.a.O., S. 59.

[4] Vgl. Kosiol, E.: Modellanalyse als Grundlage ..., a.a.O., S. 320.

[5] Vgl. Kosiol, E.: Modellanalyse als Grundlage ..., a.a.O., S. 321.

auf Beziehungen zwischen quantitativ bestimmbaren Größen beschränkt, während die Aussagenlogik sich auf Beziehungen zwischen beliebigen Objekten ausdehnen läßt."[1] Für die Tatsache, "daß sich alle nicht oder nicht hinreichend genau quantifizierbaren betrieblichen Vorgänge einer Modellbetrachtung entziehen"[2], ist diese Erkenntnis bedeutsam. Hier kann durch die aussagenlogischen Programme eine nützliche Erweiterung der Unternehmensforschung geschaffen werden. Dennoch ist die aussagenlogische Programmierung nur ein Anfangspunkt. Ihrer Kalkülisierung sind relativ enge Grenzen gesetzt. Diese überwindet die BOOLEsche Programmierung, indem sie systematische Lösungsverfahren auch für numerische Probleme bereitstellt. In diesem Zusammenhang wird dann von PSEUDO-BOOLEschen Programmen gesprochen[3].

BOOLEsche bzw. PSEUDO-BOOLEsche Programme sind eindeutig mathematische, wenn auch einer besonderen Art. Sie gehören zur Gruppe der "abstrakt-symbolischen"[4] Modelle. Für das BOOLEsche Programm sind alle relevanten Komponenten des Entscheidungsproblems mit ihren Abhängigkeiten zu ermitteln. Die im Hinblick auf das Ziel wesentlichen Größen werden in eine Menge von geordneten Symbolen abgebildet, die Wirkzusammenhänge in BOOLEsche Verknüpfungen transformiert. Das Ergebnis dieser Zuordnung sind BOOLEsche Programme, ihre Lösungen werden durch geeignete Operationen ermittelt und durch Wiedereinsetzung realer Daten fixiert.

Beachtet werden muß jedoch, daß nur dann von einem BOOLEschen Entscheidungsmodell gesprochen werden sollte, wenn

---

[1] Klaus, Georg: Moderne Logik ..., a.a.O., S. 113.

[2] Angermann, A.: Entscheidungsmodelle, a.a.O., S. 14.

[3] Vgl. Ivanescu, Petru L.: Pseudo Boolean Programming and Applications, Lecture Note in Mathematics No. 9. Berlin - Göttingen - Heidelberg 1965 und die dort angegebene Literatur.

[4] Kosiol, E.: Betriebswirtschaftslehre und Unternehmensforschung, in ZfB 34 Jg. 1964, S. 754.

unter beliebigen Optimalitätskriterien Lösungen des abgebildeten Problems angegeben werden können. Aussagenlogische Formulierungen haben keine Lösungen im üblichen Sinn. Sie bieten demgegenüber den Vorteil, komplexe, unüberschaubare Problemstrukturen auf wenige, psychologisch eingängige Aussagen zurückführen zu können.

ZWEITES KAPITEL

DAS SYSTEM DER AUSSAGENLOGISCHEN PROGRAMMIERUNG

21. Aussagenlogische Grundzüge

211. Allgemeine Grundlagen

Als die wesentlichen Kennzeichen, welche die algebraische von der klassischen Logik unterscheiden, sind die Symbolisierung, Formalisierung sowie die Kalkülisierung und Axiomatisierung genannt worden. Innerhalb der algebraischen Logik sind Aussagen-, Prädikaten-, Klassen-, Relationenkalküle und andere zu unterscheiden[1]. Die Aussagenkalküle beschränken sich auf die formalisierte Darstellung und Verknüpfung von zweiwertigen Aussagen, d.h. von Behauptungen über Sachverhalte, die generell nur zweier Zustände fähig sind. Die aussagenlogischen Sätze sind grundsätzlich losgelöst vom konkreten Inhalt, sprachliche Ungenauigkeiten sind ausgeschlossen. Dadurch ist die neue Logik der klassischen "an Strenge und Fruchtbarkeit bei weitem überlegen."[2] Mit Hilfe der Aussagenkalküle ist die "erfolgreiche Inangriffnahme von Problemen möglich, bei denen das rein inhaltliche

---

[1] Zu den verschiedenen Logikkalkülen vgl. Bochenski, J.M.; Menne, A.: Grundriß ..., a.a.O., S. 25 ff.

[2] Bochenski, J.M.; Menne, A.: Grundriß ..., a.a.O., S. 12. Siehe auch Lindsay, F.A.: New Techniques for ..., a.a.O., S. 58 f.: "As a language of reasoning, it has a great advantage over verbal language in that it is designed to be rigorously free of contradiction and ambiguity."

Denken prinzipiell versagt."[1] Darin, daß die algebraische Logik von konkreten Inhalten und sprachlichen Formulierungen vollkommen abstrahiert und nur die logische Struktur der Aussagen berücksichtigt, ist die Möglichkeit begründet, aussagenlogische Ansätze rein mechanisch zu bearbeiten. Wenn FISCHL also den Hauptmangel dieser Kalküle darin sieht, daß alle Brücken zum wirklichen Leben abgebrochen[2] sind, verkennt er den eigentlichen Sinn der logischen Algebra. Wesentlich für die Aussagenkalküle ist eine spezielle Eigenschaft, der Wahrheitswert der einzelnen Aussagen, d.h. relevant ist allein die Frage, ob der einzelne Satz eines Kalküls wahr oder falsch ist. Jede Aussage besitzt einen dieser beiden Wahrheitswerte und man kann jeweils eine Klasse der wahren und der falschen Aussagen fixieren. Unter Aussage wird dabei jede Äußerung verstanden, von der es logisch sinnvoll ist zu behaupten, daß sie wahr oder falsch ist. Nur diese Alternation ist möglich, andere Entscheidungen sind unzulässig.

Man unterscheidet im Aussagenkalkül zwischen einfachen und zusammengesetzten Aussagen. Hier interessieren besonders die Aussagenkombinationen. Zusammengefaßt werden die einzelnen Aussagen durch logische Verknüpfungen wie "und", "oder", "weder - noch" usw. Diese Verknüpfungen, die ebenso wie die Aussagen durch Symbole dargestellt werden, haben in allen Kalkülen die gleiche Bedeutung. Man nennt sie deshalb die Konstanten des aussagenlogischen Kalküls. Dazu zählen auch die Wahrheitswerte wahr und falsch. Die Aussagen selbst, welche durch die Konstanten verbunden werden und deren realer Hintergrund von Fall zu Fall wechseln kann, werden als Variable bezeichnet. Die angesprochenen Verknüpfungen können beliebig oft und in allen möglichen Kombinationen in einem Kalkül auftreten. Dabei muß beachtet werden,

---

[1] Hilbert, D.; Ackermann, W.: Grundzüge der theoretischen Logik, a.a.O., Einleitung.

[2] Fischl, J.: Logik ..., a.a.O., S. 152.

daß die elementaren Verknüpfungen durch Klammern klar voneinander getrennt werden. Klammern werden als Hilfszeichen definiert. Die aussagenlogischen(oder Wahrheits-) Funktionen bestehen aus Konstanten, Variablen und Hilfszeichen. Sie können nach bestimmten Vorschriften umgeformt - vereinfacht - werden (Kalkülisierung). Die Axiomatisierung kennzeichnet das Bestreben, alle irgendwie vorkommenden Kombinationen von Verknüpfungen durch einige wenige Grund- oder Standardausdrücke darzustellen und alle Verknüpfungsgesetze aus möglichst wenig unbewiesenen Grundsätzen herzuleiten[1].

212. Das "Rechnen" mit logischen Aussagen

Die Verknüpfungsgesetze bestimmen - wie zu zeigen sein wird - bei gegebenen Wahrheitswerten der Elementaraussagen auf eindeutige Weise den Funktions- oder Wahrheitswert eines komplexen logischen Ausdrucks. Den einzelnen Funktionen, in welche die Variablen mit den Werten 1 (d.h. wahr, ja, Strom, zu usw.) und 0 (falsch, nein, kein Strom usw.) eingehen, entsprechen also bestimmte 0,1 Ausgangszuordnungen. Da jede Variable mit 0 oder 1 vorbestimmt werden kann, sind bei n Variablen 2^n Eingänge und $2^{(2^n)}$ Funktionen fixiert. Für die zweiwertige logische Algebra ergeben sich 4 Eingangsmöglichkeiten und 16 Funktionen . Diese sind in nachstehender Tabelle zusammengefaßt. Die Tabelle ist zur besseren Übersicht entsprechend der binären Addition aufgebaut, d.h. die 0,1 Zuordnungen für die Eingänge (I, II, III, IV) und Ausgänge (1, ..., 16) erhält man durch binäre Addi-

[1] z.B. auf die sogenannte PEIRCEfunktion und die SHEFFERfunktion. Vgl. hierzu Bochenski, J.M.; Menne, A.: Grundriß ..., a.a.O., S. 28 f.

tion, jeweils oben rechts beginnend[1]. Es muß noch darauf hingewiesen werden, daß die Symbole für die einzelnen logischen Operationen in der Literatur sehr uneinheitlich sind. Aus technischen Gründen wird hier die allgemeine arithmetische Bezeichnungsweise vorgezogen, wobei immer bedacht werden muß, daß sie sich auf die Gesetze der logischen Algebra bezieht[2].

Die Ausgangswerte ergeben sich in der angegebenen Reihenfolge, wenn jede Operation nacheinander auf die vier Eingangswertkombinationen angewandt wird.

---

[1] Vgl. Zum Rechnen im binären (auch dualen) Zahlensystem
Sommer, Friedrich: Mathematik für Studenten der Wirtschaftswissenschaften, Berlin - Göttingen - Heidelberg, 1962, S. 5 f. und
Kemeny, John G.; Snell, J. Laurie und Thompson, Gerald,L.: Einführung in die Endliche Mathematik, Ludwigshafen 1963, S. 76 ff.
Zu Tabellen vgl. v. Freytag-Löringhoff: Logik ..., a.a.O., S. 157.
Taschenbuch der Nachrichtenverarbeitung ..., a.a.O., S. 105.
Bochenski, J.M.; Menne, A.: Grundriß ..., a.a.O., S. 26, 32.

[2] Zusammenstellung der gebräuchlichsten Symbole

	vorwiegend in technischer	anglo amerikanischer	mathematischer Literatur	hier
Negation	$\overline{x}$	x'	$\sim x$, $\neg x$	$\overline{x}$
Konjunktion	$x_1 \& x_2$	$x_1 \cdot x_2$	$x_1 \wedge x_2$, $x_1 \wedge x_2$	$x_1 x_2$
Disjunktion	$x_1 \vee x_2$	$x_1 + x_2$	$x_1 \vee x_2$, $x_1 \vee x_2$	$x_1 + x_2$

Die logischen Grundverknüpfungen bei zwei Variablen

Eingang für			Operation	Nr.	Ausgang für				sprachliche Entsprechung	Bezeichnung
	x_1	x_2			I.	II.	III.	IV.		
I.	0	0	0	1	0	0	0	0	nie	Nullfunktion
II.	0	1	$x_1 x_2$	2	0	0	0	1	und	Konjunktion
III.	1	0	$x_1 \bar{x}_2$	3	0	0	1	0	und	Konjunktion
IV.	1	1	x_1	4	0	0	1	1	-	-
I.	0	0	$\bar{x}_1 x_2$	5	0	1	0	0	und	Konjunktion
II.	0	1	x_2	6	0	1	0	1	-	-
III.	1	0	$x_1 \not\equiv x_2$	7	0	1	1	0	oder (aut)	Antivalenz
IV.	1	1	$x_1 + x_2$	8	0	1	1	1	oder (vel)	Disjunktion
I.	0	0	$\bar{x}_1 \bar{x}_2$	9	1	0	0	0	weder - noch	PEIRCEfunktion
II.	0	1	$x_1 \equiv x_2$	10	1	0	0	1	gleich	Äquivalenz
III.	1	0	$\bar{x}_1$	11	1	0	1	0	-	-
IV.	1	1	$x_1 + \bar{x}_2$	12	1	0	1	1	wenn - dann	Implikation
I.	0	0	$\bar{x}_1$	13	1	1	0	0	-	-
II.	0	1	$\bar{x}_1 + x_2$	14	1	1	0	1	wenn - dann	Implikation
III.	1	0	$x_1 \mid x_2$	15	1	1	1	0		SHEFFERfunktion
IV.	1	1	1	16	1	1	1	1	immer	Einsfunktion

Tabelle 1

Diese Tabelle enthält alle sogenannten Wahrheitstafeln, die für die einzelnen logischen Funktionen gebildet werden können.

Die Funktionen Nr. 1 und Nr. 16 stellen Konstante dar, während Nr. 4, 6, 11 und 13 nur von einer einzigen Variablen abhängen und als trivial zu kennzeichnen sind. Hier interessieren hauptsächlich die folgenden Verknüpfungen:

(1) Die KONJUNKTION - auch logisches Produkt genannt - ist dann und nur dann wahr, wenn alle Argumente (Variablen) wahr sind, oder ein Ereignis tritt nur ein, wenn alle Bedingungen erfüllt sind.

SCHWEITZER hat versucht, aufbauend auf später näher zu besprechende amerikanische Ansätze, die Konjunktion neben anderen logischen Operationen "in das Grundkonzept der Strukturierung des Aktionsgefüges einzubauen".[1] An einem Stückprozeß erklärt er die Und-Funktionen, indem er die erforderliche Absatzreife als Funktion aller notwendigen Arbeitsgänge definiert, welche erst eintritt, wenn letztere ausnahmslos vollzogen sind[2].

Besser läßt sich die konjunktive Verknüpfung betrieblicher Sachverhalte zeigen, wenn man das Unternehmen als kybernetisches Regelsystem auffaßt: Nur wenn alle Einflußfaktoren den geforderten Stellwert realisieren (z.B. keine negative Abweichung in Form der Illiquidität), ergibt sich der angestrebte Extrem-(System-)wert. Die Konjunktion entspricht dem mengentheoretischen Durchschnitt.

Definiert man x_1, x_2, x_3 ... als beliebige Aussagen und M als die Menge ihrer logischen Möglichkeiten, X_1, X_2,

---

[1] Schweitzer, Marcell: Probleme der Ablauforganisation in Unternehmungen, Berlin 1964, S. 7.

[2] Schweitzer, M.: Probleme ..., a.a.O., S. 77.

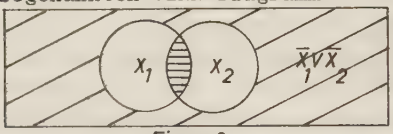
X_3 ... als die Teilmengen von M, für welche die betreffenden Aussagen wahr sind und bezeichnet die Mengen X_1, X_2, X_3 ... als die Wahrheitsmengen von x_1, x_2, x_3, dann ist z.B. die Wahrheitsmenge von x_1x_2 gleich dem Durchschnitt $X_1 \cap X_2$.

Es gilt somit in der Wahrheitstafel

x_1	x_2	x_1x_2
0	0	0
0	1	0
1	0	0
1	1	1

Tabelle 2

bzw. im sogenannten VENN-Diagramm[1]



Figur 2

(2) Die Disjunktion - auch logische Summe oder Adjunktion genannt - entspricht dem sprachlichen "oder" im Sinne von "oder auch" (lat. vel) und ist dann wahr, wenn von den Argumenten der Funktion wenigstens eins wahr ist.

(3) Die NEGATION stellt den Sonderfall einer Funktion mit nur einer Variablen dar. Sie bildet aus einer wahren Aussage eine falsche und umgekehrt. Entsprechend führt doppelte Negation auf den Eingangswert zurück. Mengentheoretisch entspricht der Negation das Komplement. Die Bedingung (das Argument) ist also so formuliert, daß das Ereignis nur dann eintritt (die Funktion nur dann

[1] Andere Darstellungsformen - besonders auch zur Vereinfachung der Ausdrücke sind das sogenannte KARNAUGH- und VEITCH-Diagramm. Vgl. dazu Weyh, U.: Elemente der ..., a.a.O., S. 74 und die dort angegebene Literatur.

wahr wird), wenn sie (es) verneint wird.

(4) Die IMPLIKATION entspricht etwa dem sprachlichen wenn - dann, wobei jedoch zu beachten ist, daß allein das "formale Verhältnis der hinreichenden Bedingung"[1] wiedergegeben wird und keinerlei sprachliche Inhalte kausal oder konditional verknüpft werden. Wenn x_2 durch x_1 (Implikant) impliziert wird, bedeutet dies, daß x_2 (Konklusion) aus x_1 logisch abgeleitet werden kann. Die Implikation ist nur dann falsch, wenn die Konklusion falsch ist, in allen anderen Fällen ist sie wahr.

Stellt man folgende Wahrheitstabelle

x_1	x_2	$\overline{x}_1$	$x_1 \longrightarrow x_2$	$\overline{x}_1 + x_2$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	0
1	1	0	1	1

Tabelle 3

zusammen, so erkennt man, daß die implikative Verknüpfung $x_1 \longrightarrow x_2$ und die disjunktive Verknüpfung mit einem negierten Element die gleiche Wahrheitsmenge haben, beide Ausdrücke sind äquivalent, d.h. $x_1 \longrightarrow x_2 = \overline{x}_1 + x_2$. Hier zeigt sich bereits die axiomatische Grundlage der logischen Algebra: Alle Verknüpfungen lassen sich durch Kürzungen bzw. Umformungen auf sogenannte Normalformen reduzieren. Insbesondere unterscheidet man in der Vielfalt der äquivalenten aussagenlogischen Darstellungsmög-

[1] Bochenski, J.M.; Menne, A.: Grundriß ..., a.a.O., S. 27/28.

lichkeiten zwei Normalformen, in die die logischen Funktionen überführt werden können, die konjunktive und die disjunktive Normalform. Beide bauen nur auf konjunktiven bzw. disjunktiven Verknüpfungen und der Negation auf. Außerdem wird z.B. für die disjunktive Normalform gefordert, daß in einem negierten Klammerausdruck weder negierte Aussagen noch weitere disjunktive bzw. konjunktive Verknüpfungen auftreten und im Klammerbereich einer Konjunktion keine disjunktiven Verknüpfungen auftreten. Entsprechendes gilt für die konjunktive Normalform.

Grundlage für die Umformung der logischen Ausdrücke ist die Kenntnis folgender Gesetze und Rechenregeln[1]. Ein aussagenlogisches System, welches diese Axiome erfüllt, ist damit gleichzeitig als BOOLEsche Algebra definiert.

I. Wenn x_1 und x_2 Elemente eines Systems sind, gibt es ein Element x_3, so daß $x_1 \cdot x_2 = x_3$ ist.

II. Wenn x_1 und x_2 Elemente sind, gibt es ein Element x_4, so daß $x_1 + x_2 = x_4$ ist.

III. Es gibt ein Element 0 und wenn x_1 irgendein Element ist, gilt $x_1 + 0 = x_1$.

IV. Es gibt ein Element 1 und wenn x_1 irgendein Element ist, gilt $x_1 \cdot 1 = x_1$.

V. Wenn x_1 Element des Systems ist, so existiert ein Element $\overline{x}_1$, so daß $x_1 \cdot \overline{x}_1 = 0$ und $x_1 + \overline{x}_1 = 1$ ist.

VI. Wenn x_1 und x_2 Elemente sind, gilt $x_1 \cdot x_2 = x_2 \cdot x_1$.

VII. Wenn x_1 und x_2 Elemente sind, gilt $x_1 + x_2 = x_2 + x_1$.

---

[1] Vgl. Hilbert, D. und Ackermann, W.: Grundzüge ..., a.a.O., S. 12.
Schmidt, Arnold: Mathematische Gesetze ..., a.a.O., S. 13 ff.
Dwinger, Ph.: Introduction to Boolean Algebras, Würzburg 1961, S. 2 ff.

VIII. Wenn x_1, x_2 und x_3 Elemente sind, so gilt
$x_1 + (x_2 \cdot x_3) = (x_1 + x_2) \cdot (x_1 + x_3)$.

IX. Wenn x_1, x_2 und x_3 Elemente sind, so gilt
$x_1 \cdot (x_2 + x_3) = x_1 \cdot x_2 + x_1 \cdot x_3$[1].

X. Wenn x_1 und x_2 Elemente sind, so gilt $x_1 + x_1x_2 = x_1$ und $x_1 \cdot (x_1 + x_2) = x_1$.

XI. Es gibt Elemente x_1 und x_2, so daß $x_1 \neq x_2$.

I. und II. stellen Verknüpfungen, V. die Negation dar. Die Axiome III. und IV. beziehen sich auf keine logische Aussage (Gesetz der Null) und die Gesamtheit aller logischen Aussagen (Gesetz der Eins). VI. und VII. stellen das kommutative, VIII. und IX. das distributive Gesetz dar. X. schließlich bezieht sich auf die Absorption von Gliedern und XI. verlangt für einen BOOLEschen Verband, daß außer der Negativen zwei Elemente existieren, die verschieden sind. Selbstverständlich gibt es neben der zweiwertigen BOOLEschen Algebra noch andere logische Systeme die drei- oder mehrwertig sind. Da hier ein Formalismus entwickelt werden soll, mit dem oekonomische Alternativentscheidungen gefällt werden sollen, bleiben mehrwertige Systeme außer Betracht.

Die einzelnen Gesetze lassen sich leicht mit Hilfe der Wahrheitstafeln überprüfen, die für die dargestellten Äquivalenzen gleiche Werte aufweisen, z.B. wie in Spalte 7 und 8 folgender Tabelle für das distributative Gesetz[2].

---

[1] Wie in der Arithmetik bindet auch hier " · " stärker als " + ", deshalb kann die Klammer an dieser Stelle entfallen.

[2] Andere ausführliche Beispiele für die schrittweise Vereinfachung komplizierter aussagenlogischer Ausdrücke bei Klaus, G.: Moderne Logik ..., a.a.O., S. 103 ff.

x_1	x_2	x_3	$x_1 \cdot x_2$	$x_1 \cdot x_3$	$x_2 + x_3$	$x_1 \cdot (x_2 + x_3)$	$x_1 \cdot x_2 + x_1 \cdot x_3$
0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0
0	1	0	0	0	1	0	0
0	1	1	0	0	1	0	0
1	0	0	0	0	0	0	0
1	0	1	0	1	1	1	1
1	1	0	1	0	1	1	1
1	1	1	1	1	1	1	1

Tabelle 4

Die Verknüpfungen sind nicht auf zwei unabhängige Variable beschränkt, vielmehr können beliebig viele Elemente, die natürlich alle boolesch sind, d.h. 0 oder 1, miteinander verknüpft werden. Konjunktionen oder Disjunktionen aller Variablen eines Systems werden als Vollkonjunktionen oder "Minterme" bzw. Volldisjunktionen oder "Maxterme" bezeichnet. Normalformen aller Variablen heißen vollständig oder ausgezeichnet.

22. Die "Aussagenlogische" Programmierung

221. Die aussagenlogische Formulierung von Entscheidungsproblemen

Praktische Entscheidungsprobleme sind häufig derart kompliziert, daß ihre verbale Beschreibung die grundlegenden Zusammenhänge nicht oder nur unvollkommen wiedergibt. Optima-

le Entscheidungen sind dann schon deshalb ausgeschlossen, weil aus dem konkreten Entscheidungsproblem nicht alle logisch möglichen Alternativen entwickelt werden können, denn jede Entscheidungssituation kann durch eine vollständige Liste aller Alternativen gekennzeichnet werden. Die Erarbeitung dieser Alternativen ist der notwendige erste Schritt im Entscheidungsprozeß[1]. Ist sie nicht möglich, trifft der Entscheidende seine Wahl aufgrund eines unvollständigen Kollektivs von Alternativen, das mehr oder minder willkürlich durch Vorentscheidungen (Informationsentscheidungen) oder sprachliche Zufälligkeiten fixiert ist. Vielfach bescheidet sich der Aktor auch bewußt mit einer Teilmenge der möglichen Alternativen und verzichtet aus Kosten- oder Zeitgründen auf eine weitergehende Aufhellung seines oekonomischen Horizonts, d.h. auf eine weitere Annäherung seines subjektiven Informationsstandes an den objektiv möglichen[2].

Mit Hilfe aussagenlogischer Formulierungen können derartige Schwierigkeiten teilweise vermieden werden. Dazu werden die einzelnen Entscheidungskomponenten in eine Symbolsprache übertragen und ihre Zusammenhänge durch logische Verknüpfungen dargestellt. Die derart gewonnenen Aussagenkalküle sind entsprechend den aussagenlogischen Gesetzen so umzuformen, daß alle möglichen Entscheidungsalternativen entwickelt werden. Da - wie bereits erwähnt - derartige Programme keine Lösung im herkömmlichen arithmetischen Sinne haben, sind die Alternativen als Ergebnis der logischen

---

[1] Vgl. Gäfgen, Gerard: Theorie der wirtschaftlichen Entscheidung, Tübingen 1963, S. 105. Als zweiter Schritt ist die erschöpfende Beschreibung aller Konsequenzen der verschiedenen Alternativen zu nennen.

[2] Vgl. zu Informationsgrad bzw. Informationsstand Wittmann, Waldemar: Unternehmung und unvollkommene Information, Köln und Opladen 1959, S. 23 ff. und S. 83 ff.

Operationen in einer "psychologisch zugänglichen Form"[1] darzustellen. Diese ist die disjunktive Normalform, welche unmittelbar verbal interpretiert werden kann. Folgendes Beispiel soll das beschriebene Vorgehen erläutern.

Ein Betrieb steht beim Aufbau seiner Absatzorganisation vor folgendem Entscheidungsproblem:

1. Sollen Fachhändler direkt beliefert werden einschließlich der Kaufhäuser (Symbol F) oder die Warenhäuser und der Lebensmitteleinzelhandel (L) oder der Sortiments- bzw. Spezialgroßhandel (G)? Die gleichzeitige Inanspruchnahme mehrerer Absatzwege ist ausgeschlossen[2].

2. Der Absatz über den Fachhandel macht den Aufbau eines eigenen Vertreterstammes (V) erforderlich sowie direkte Werbung (W) und Preisbindung der Produkte (P).

3. Dagegen läßt der Absatz über den Großhandel, über die Warenhäuser und den Lebensmitteleinzelhandel eine lückenlose Preisbindung nicht zu.

4. Bei Belieferung des Großhandels soll nicht direkt geworben werden, notwendig sind jedoch ebenfalls firmeneigene Vertreter.

Diese Problemstruktur wird in ein aussagenlogisches Programm übertragen. Dafür gilt:

I. Für die Entscheidungs-(Aussagen-)variablen F, L, G, V, W und P, allgemein x_i, wird gesetzt x_i, wenn die positive Entscheidung zulässig ist, $\bar{x}_i$ sonst.

---

[1] Adam, A.: Aussagenlogische Programmierung, in: Ufo 3. Bd. 1959, S. 136 ff. (Ufo-Beratungsdienst).
Adam, A.: Systematische Datenverarbeitung bei der Auswertung von Versuchs- und Beobachtungsergebnissen, Würzburg 1963, S. 44 f.

[2] Derartige Entscheidungsprobleme sind häufig anzutreffen. Zu Einzelheiten und Begründung vgl. die einschlägige absatzwirtschaftliche Literatur z.B. Gutenberg, Erich: Grundlagen der Betriebswirtschaftslehre, 2. Bd. Der Absatz 5. Aufl. Berlin - Göttingen - Heidelberg 1962.

II. Da alle vier Bedingungen als logische Beschränkungen des Problems erfüllt sein müssen, wenn die einzelnen Alternativen zulässig sein sollen, ergeben sich vier "Beschränkungsgleichungen."

III. Das Ziel besteht darin, möglichst viele (entspricht einer Extremwertbedingung), d. h. in diesem Fall alle zulässigen Alternativen des Entscheidungsproblems zu entwickeln. Diese erhält man notwendig, wenn unter Beachtung der Beschränkungen die vollständige disjunktive Normalform der Aussagenfunktion gebildet wird.

Die sogenannte Aussagenfunktion stellt sich dar als die konjunktive Verknüpfung der vier Beschränkungen:

$$f = B_1 \cdot B_2 \cdot B_3 \cdot B_4.$$

Diese Aussagenfunktion wird entsprechend dem unter III. angegebenen Ziel solange umgeformt - entwickelt - bis alle bei Beachtung der vier Beschränkungen möglichen Disjunktionsglieder vollständig sind. Die unbesetzten Stellen in den Beschränkungsgleichungen sind dabei mit der Ziffer 1 zu versehen[1]. Anschließend sind die Beschränkungen logisch auszumultiplizieren, um die ausgezeichnete disjunktive Normalform zu erhalten.

Man hat somit:

B_1 : $F\bar{L}\bar{G}111 + \bar{F}L\bar{G}111 + \bar{F}\bar{L}G111$

B_2 : $\bar{F}11111 + 111VWP$

B_3 : $1\bar{L}\bar{G}111 + 11111\bar{P}$

B_4 : $11\bar{G}111 + 111V\bar{W}1$

[1] 1 muß hier als Ziffer, nicht als logischer Wert für wahr, d.h. sprachlich als neutral oder indifferent definiert werden. Die Aussagenvariable kann an dieser Stelle entsprechend $x_1 + \bar{x}_1 = 1$ beliebig einen der beiden Wahrheitswerte annehmen.

Daraus ergibt sich für die konjunktive Verknüpfung von

$$B_1 \cdot B_2 = (F\bar{L}\bar{G}111 + \bar{F}L\bar{G}111 + \bar{F}\bar{L}G111)(\bar{F}11111 + 111VWP)$$
$$= \bar{F}L\bar{G}111^{1} + \bar{F}\bar{L}G111 + F\bar{L}\bar{G}VWP + \bar{F}L\bar{G}VWP + \bar{F}\bar{L}GVWP$$

und weiter

$$(B_1 \cdot B_2) \cdot B_3 = (\bar{F}L\bar{G}111 + \bar{F}\bar{L}G111 + F\bar{L}\bar{G}VWP + \bar{F}L\bar{G}VWP + \bar{F}\bar{L}GVWP)$$
$$\cdot(1\bar{L}\bar{G}111 + 11111\bar{P})$$
$$= F\bar{L}\bar{G}VWP + \bar{F}L\bar{G}11\bar{P} + \bar{F}\bar{L}G11\bar{P}$$

und $(B_1B_2B_3)B_4 = (F\bar{L}\bar{G}VWP + \bar{F}L\bar{G}11\bar{P} + \bar{F}\bar{L}G11\bar{P})(11\bar{G}111 + 111V\bar{W}1)$
$$= F\bar{L}\bar{G}VWP + \bar{F}L\bar{G}11\bar{P} + \bar{F}L\bar{G}V\bar{W}\bar{P} + \bar{F}\bar{L}GV\bar{W}\bar{P}$$

Das zweite Disjunktionsglied $\bar{F}L\bar{G}11\bar{P}$ spaltet sich auf in

$$\bar{F}L\bar{G}VW\bar{P} \quad \bar{F}L\bar{G}V\bar{W}P \quad \bar{F}L\bar{G}\bar{V}W\bar{P} \quad \bar{F}L\bar{G}\bar{V}\bar{W}\bar{P}.$$

Das zweite Produkt entfällt, da bereits vorhanden und die Aussagenfunktion nicht wertschweifig (irredundant) sein darf. Irredundant ist eine aussagenlogische Funktion, die keine überflüssigen Variablen und keine überflüssigen Glieder enthält[2].

Es verbleiben somit sechs Alternativen (zulässige Entscheidungen):

1. $F\bar{L}\bar{G}VWP$
2. $\bar{F}L\bar{G}V\bar{W}\bar{P}$
3. $\bar{F}\bar{L}GV\bar{W}\bar{P}$
4. $\bar{F}L\bar{G}VW\bar{P}$
5. $\bar{F}L\bar{G}\bar{V}W\bar{P}$
6. $\bar{F}L\bar{G}\bar{V}\bar{W}\bar{P}$,

welche unmittelbar interpretiert werden können.

---

[1] Das erste Glied entfällt, da $F \cdot \bar{F} = 0$.

[2] Vgl. Quine, W.V.: The Problem of Simplifying Truth Functions, in AMM, Vol. 59 1962, S. 522.

Selbstverständlich kann an dieser Stelle der Einwand vorgebracht werden, daß derartige Probleme rein gedanklich zu lösen sind. Das ist aber bereits bei Entscheidungssituationen mit mehr als zehn Variablen nahezu unmöglich, schon bei wenigen logischen Beschränkungen: Das sind die oben angesprochenen Fälle, in denen die Suche nach allen Alternativen von vornherein aufgegeben wird.

Die aussagenlogische Programmierung ermöglicht hier ein systematisches Vorgehen, das mit Sicherheit auf mechanische Weise (programmierbar!) alle Alternativen liefert. Wichtig ist in diesem Zusammenhang die Feststellung, daß mit einer Ausnahme alle bislang bekanntgewordenen Anmerkungen über die Bedeutung der algebraischen Logik für die Unternehmensforschung sich auf diese Art von Anwendungen beschränken[1]. Die algebraische Logik wird dann begriffen "as a tool for analysis of a practical situation", wobei "its value lies in its ability to reduce complex problems to cleare, concise terms."[2] Wie aber zu zeigen sein wird, kann die BOOLEsche Algebra für die Unternehmensforschung weit mehr leisten als ein "Werkzeug zur Analyse."

---

[1] Das gilt z.B. für Cushen, W.E.: Symbolic Logic ..., a.a.O., S. 187. Vgl. dazu Kulhavy, Ernest: Operations Research, Wiesbaden 1963, S. 85, der sich auf die Wiedergabe der Gedanken von CUSHEN beschränkt.
Weiter Smidt, Robert M.; Reis, Irvin L.: Symbolic Logic and Plant Location, in: JIE, Vol. XIV No. 1 1963, S. 18 ff.
Ackerman, Sanford S.: Symbolic Logic: A Summary of the Subject and its Application to Industrial Engineering, in: JIE, Vol. VIII. 1957, No. 5, S. 293 ff.

[2] Ackerman, S.S.: Symbolic Logic ..., a.a.O., S. 299.

222. Das Modell von AKERS und FRIEDMAN als aussagenlogisches Programm

2221. Die Fragestellung

Einen eigenständigen und interessanten, zudem den einzigen auch in Deutschland diskutierten[1] Beitrag zur Anwendung der logischen Algebra in der Unternehmensforschung, der über die im vorigen Abschnitt erwähnten Beschränkungen hinausgeht, haben AKERS und FRIEDMAN 1955 für das Problem der Ablaufplanung vorgelegt[2].

Innerhalb der Unternehmensforschung muß man die Konstruktion von Ablaufplanungsmodellen zu jenen Problemen zählen, welche "die höchsten Anforderungen stellen."[3] Besonders schwierig sind hierbei die Fragen der Maschinenbelegungs- und Ablaufplanung bei Betrieben der Werkstatt- und Einzelfertigung. Treffend wird dieser Problemkreis mit dem von GUTENBERG geprägten Ausdruck vom "Dilemma der Ablaufplanung"[4] gekennzeichnet, welches dadurch entsteht, daß mit der Minimierung der Leerzeiten und der Optimierung der Kapazitätsauslastung der Aggregate zwei widerstrebende Ziele verfolgt werden. Derartige Probleme sind ebenso häufig wie vielgestaltig.

---

[1] z.B. von Schweitzer, Marcell: Probleme der Ablauforganisation in Unternehmungen, Berlin 1964 und Hoss, Klaus: Fertigungsablaufplanung mittels operationsanalytischer Methoden, Würzburg - Wien 1965.

[2] Auf der 3. Jahrestagung der amerikanischen Gesellschaft für Unternehmensforschung (Operations Research Society of American) am 4.6.1955. Veröffentlicht in: Operations Research. The Journal of the Operations Research Society of America (JORSA) Vol. 3 1955, No. 4, S. 429 ff. unter dem Titel: A non-numerical Approach to production scheduling Problems.

[3] Churchman, C. West; Ackoff, Russell L.; Arnoff, Leonard E.: Operations Research. Eine Einführung in die Unternehmensforschung, Wien und München 1961, S. 430.

[4] Gutenberg, Erich: Grundlagen der Betriebswirtschaftslehre 1. Bd. Die Produktion, 6. Aufl. Berlin - Göttingen - Heidelberg 1962, S. 157.

Die Anzahl der zusammenwirkenden Maschinen, die der erforderlichen Arbeitsoperationen und die technischen Bedingungen (z.B. Überspringen von Maschinen, konstante oder wechselnde Operationenfolge usw.) können in allen möglichen Kombinationen auftreten: "Zu der Komplexion des Gesamtmodells durch Kombination beliebig vieler Variabler und mehrerer Optimalbedingungen ist die "Komplexion" durch mehrere gegensätzliche Optimierungsziele getreten."[1] Entsprechend gibt es viele, größtenteils jedoch ganz spezielle Ansätze für die einzelnen Probleme.[2] Die Autoren bedienen sich dabei sowohl analytischer als auch graphischer Darstellungsweisen. Bislang sind aber nur für die relativ unproblematischen Fälle brauchbare Entscheidungsregeln entwickelt[3] worden, "eine allgemeine Lösung des Problems, einen optimalen Maschinenbelegungsplan für m Maschinen und n Aufträge zu berechnen, gibt es jedoch noch nicht."[4] Durch den neuartigen Ansatz von AKERS und FRIEDMAN ist die Diskussion um Ablaufplanungsmodelle jedoch sehr belebt worden[5]. Weil dieses Mo-

---

[1] Hoss, K.: Fertigungsablaufplanung ..., a.a.O., S. 185. Eine ausführliche Darstellung dieser "Komplexion betrieblicher Planungsmodelle bringt Kern, W.: Gestaltungsmöglichkeit und Anwendungsbereich betriebswirtschaftlicher Planungsmodelle, in: ZfhF 14. Jg. 1962, S. 167 ff., insb. S. 169 und 172.

[2] Vgl. hierzu Hoss, K.: Fertigungsablaufplanung ..., a.a.O., der in seiner Darstellung ziemlich vollständig ist.

[3] Vgl. Churchman, C.W.; Ackoff, R.L.; Arnoff, E.L.: Operations Research ..., a.a.O., S. 430.

[4] Albach, Horst: Maschinenbelegungspläne bei Einzelfertigung, Sonderdruck aus dem Jahrbuch 1965 des Landesamtes für Forschung des Landes Nordrhein-Westfalen, Köln und Opladen, S.21.

[5] Vgl. dazu Akers, S.B.: A Graphical Approach to Production Scheduling Problems, in: JORSA, Vol. 4 1956, No. 2, S. 244 ff.
Giffler, B.; Thompson, G.L.: Algorithmus for Solving Production Scheduling Problems, in: JORSA, Vol. 8 1960 No. 4, S. 487 ff.
Giffler, B.: Scheduling General Production Systems using Scheduling Algebra, in: Naval Research Logistics Quarterly, Vol. 10. 1963 No. 3, S. 237.
Szwarc, W.: Solution of the Akers-Friedman Scheduling Problem, in: JORSA Vol. 8 1960 No. 6, S. 782 ff.
Schweitzer, M.: Probleme der ..., a.a.O.
Hoss, K.: Fertigungsablaufplanung ..., a.a.O.
Zimmermann, W.: Modellanalytische Verfahren zur Bestimmung optimaler Fertigungsprogramme, Berlin 1966, S.127 ff.

dell also auch in deutschsprachigen Interpretationen vorliegt[1], können wir uns auf eine knappe Darstellung beschränken und das Schwergewicht auf seine kritische Durchleuchtung legen.

2222. Die Modellbeschreibung

Viele Ablaufplanungsmodelle sind dadurch gekennzeichnet, daß die Reihenfolge, in der n unterschiedliche Arbeiten von m Maschinen ausgeführt werden müssen, als fest vorgegeben angenommen wird. GIFFLER-THOMPSON sprechen vom "offenen Problem"[2]. Auch AKERS und FRIEDMAN gehen von dieser Annahme aus.

Weiter gelten folgende Bedingungen:

(1) Kein Aggregat kann zur gleichen Zeit mehr als eine Operation ausführen. Operation ist die vollständige Ausführung eines Arbeitsvorganges.

(2) Alle Operationen werden sobald als möglich ausgeführt.

(3) Jede einmal begonnene Operation wird beendet.

(4) Jede spezielle Maschine ist nur einmal vorhanden.

(5) Die Bearbeitungszeit ist unabhängig von der Reihenfolge der Operationen.

(6) Transportzeiten zwischen Maschinen und Lägern bleiben außer Betracht.

---

[1] Vgl. Schweitzer, M.: Probleme der ..., a.a.O., S. 75 ff.
Hoss, K.: Fertigungsablaufplanung ..., a.a.O., S. 92 ff.
Albach, H.: Maschinenbelegungspläne ..., a.a.O., S. 33 f.

[2] Giffler, B.; Thompson, G.L.: Algorithmus for ..., a.a.O., S. 490 ff. Dual-offenes Problem bedeutet entsprechend, daß die Reihenfolge, in der die verschiedenen Aggregate die Objekte zur Durchführung bestimmter Operationen annehmen müssen, vorgegeben ist. Sind die Maschinenfolgen (1. Fall) und die Arbeitsfolgen (2. Fall) nur teilweise fixiert, liegt ein "gemischtes Problem" vor.

Von GIFFLER-THOMPSON wird weiter vorausgesetzt, daß für jede Operation eine endliche, bekannte Zeitdauer gegeben ist. Diese Bedingung wird hier nicht gestellt, um mit AKERS und FRIEDMAN zu zeigen, "that without consideration of numerical values, it is possible to eliminate initially a great number of the possible programmes by purely non-numerical methods."[1]

Im folgenden werden bezeichnet mit a, b, c, ..., z die m Maschinen und mit 1, 2, 3, ..., n die n Arbeiten. Bestimmte Operationen sind durch die Zuordnungen: a_1, b_2, c_3,...,z_n definiert. Für die Maschinenfolgen gilt die Verknüpfung durch das Folgezeichen $\prec$ (unmittelbare Folge $\prec\prec$) z.B. $b_2 \prec c_2 \prec d_2$ oder nur $b_2 \ldots c_2 \ldots d_2$, für die Arbeitsfolgen $A_{(1\prec 2)} = A$ (d.h. auf Maschine a Arbeit 1 vor Arbeit 2) bzw. $A_{(2\prec 1)} = \overline{A}$ (Negation der Aussage A)[2]. Letztere Anordnungen sind die den Prozeß kennzeichnenden grundlegenden technischen Abhängigkeiten.

---

[1] Akers, S.B.; Friedman, J.: A non-numerical Approach ..., a.a.O., S. 429

[2] Hierzu muß vermerkt werden, daß die Bezeichnungen A bzw. $\overline{A}$ nur eindeutig sind, wenn das untersuchte Problem nur 2 Arbeiten je Maschine enthält. Bei mehreren Arbeiten müßten z.B. für $A_{(3\prec 4)}$ oder $A_{(4\prec 3)}$ weitere Indizes eingeführt werden, oder es wird durch A bzw. $\overline{A}$ nur ausgedrückt, daß es sich um die 'natürliche' Anordnung $A_{(n-1\prec n)}$ handelt oder nicht. Da wir in unseren Beispielen von 2 Arbeiten ausgehen, wollen wir die dargestellte Bezeichnungsweise wählen.

Die Lösung des Problems erfolgt in drei Schritten:

(1) Es werden alle $(n!)^m$ theoretisch möglichen Zuordnungen (Programme) ermittelt[1].

(2) Mit Hilfe der BOOLEschen Algebra werden die technisch realisierbaren Programme ausgewählt.

(3) Möglich-optimale Programme werden als Teilmenge der realisierbaren Programme ebenfalls durch logische (BOOLesche) Schlußweise fixiert.

Die Bestimmung der "gar-optimalen"[2] Programme kann im Anschluß daran, sofern Prozeßzeiten bekannt sind, mit Hilfe anderer

---

[1] Es ist zu beachten, daß der Zahl $(n!)^m$ für die Anzahl der möglichen Zuordnungen bereits eine Auswahl nach dem Rationalprinzip zugrundeliegt; es werden nur effiziente Zuordnungen berücksichtigt. Ineffiziente Anordnungen sind dadurch gekennzeichnet, daß mögliche Zeitverkürzungen, die ohne Änderung der Reihenfolge möglich sind, nicht wahrgenommen werden. Bei 2 Maschinen und 2 Arbeiten sind nur 4 Anordnungen effizient. Diese eindeutige Aussage setzt voraus, daß für die möglichen Strukturregeln, AB, $\overline{A}B$, $A\overline{B}$ und $\overline{A}\overline{B}$, jeweils nur eine, die günstigste Anordnung in Betracht gezogen wird.

[2] Zum Begriff der möglich - oder gar - (besser vielleicht tatsächlich-)optimalen Programme vgl. Hoss, K.: Fertigungsablaufplanung ..., a.a.O., S. 101.

Methoden (z.B. GANTT-Diagramme) erfolgen.

Die direkte Bestimmung der technisch realisierbaren Programme als Teilmenge der theoretisch möglichen ist angesichts der bereits bei kleinen Problemen unüberschaubaren Zahl von möglichen Zuordnungen[1] ausgeschlossen. AKERS – FRIEDMAN gehen deshalb den umgekehrten Weg und formulieren Bedingungen, durch welche nur die Programme, die sie erfüllen, zugelassen werden. Diese Bedingungen gehen von unauflösbaren Zyklen aus. Es ist leicht einzusehen, daß unabhängig von zeitlichen Überlegungen ein Programm stets dann nicht realisierbar ist, wenn hinsichtlich Maschinen und Arbeiten unverträgliche Reihenfolgen gefordert werden[2]. Es sind also Bedingungen anzugehen, die notwendig und hinreichend alle nicht realisierbaren Programme erfassen. Zyklen lassen sich für alle Programme von n Arbeiten auf m Maschinen nachweisen, für welche die Arbeitsfolgen

$$A_{(2<1)},\ B_{(3<2)},\ C_{(4<3)},\ \ldots,\ Y_{(\alpha<\alpha-1)},\ Z_{(1<\alpha)}$$

sowie die Maschinenfolgen

$$\begin{array}{rl} 1. & \ldots\ a_1 \ldots\ z_1 \ldots \\ 2. & \ldots\ b_2 \ldots\ a_2 \ldots \\ 3. & \ldots\ c_3 \ldots\ b_3 \ldots \\ & \vdots \\ \alpha-1. & \ldots\ y_{\alpha-1} \ldots\ x_{\alpha-1} \ldots \\ \alpha. & \ldots\ z_\alpha \quad \ldots\ y_\alpha \quad \ldots \end{array}$$

---

[1] etwa bei 5 Arbeiten auf 5 Maschinen $(5!)^5 = 24.883.200.000$ mögliche Zuordnungen.

[2] z.B. seien für die Arbeiten 1 und 2 die Maschinenfolgen 1. $a_1 \ldots b_1 \ldots c_1$ und 2. $b_2 \ldots a_2 \ldots c_2$ sowie die Bearbeitungsvorschrift $\overline{AB}$ gegeben. Hier besteht ein unauflösbarer Zyklus. Das Programm ist nicht realisierbar: Arbeit 1 kann nicht begonnen werden, da $a_2 < a_1$ (entsprechend $\overline{A}$), Arbeit 2 ebenfalls nicht, da $b_1 < b_2$ (aus B).

gefordert werden. Diese Annahmen führen nämlich wie folgt über:

$z_\alpha < y_2$ (entsprechend Maschinenfolge α).
$y_\alpha < y_{(\alpha-1)}$ (entsprechend den angegebenen Arbeitsfolgen)
$y_{(\alpha-1)} < x_{(\alpha-1)}$ (entsprechend Maschinenfolge $\alpha - 1$)
$x_{(\alpha-1)} < x_{(\alpha-2)}$ (entsprechend den Arbeitsfolgen)
.
.
.
$b_3 < b_2$
$b_2 < a_2$
$a_2 < a_1$
$a_1 < z_1$
$z_1 < z_\alpha$

notwendig zu der Forderung $z_\alpha < z_\alpha$. Dieser unauflösbare Zyklus bestimmt alle entsprechenden Programme als nicht realisierbar.

Zum Nachweis der Hinlänglichkeit der angegebenen Bedingungen wird von einem für $q + 1$ Operationen nicht durchführbaren Programm ausgegangen, welches für beliebige α der Maschinenfolgen (wobei $2 \leq \alpha < q + 1$) die Bedingungen erfüllt. Es läßt sich nachweisen, daß dann für alle $q + 1$ Maschinenfolgen ein Zyklus existiert und daß die entsprechenden Anordnungen in den betreffenden Programmen erscheinen.

Wenn ein beliebiges Programm für $q + 1$ Maschinenfolgen nicht realisierbar ist, muß es in jeder Folge eine Operation geben, die nicht ausgeführt werden kann, d.h. das Programm blockiert.

Z.B. seien in

1. ... a_1 ...
2. ... b_2 ...
3. ... c_3 ...

.
.
.

q+1. ... z_{q+1} :...

"blockierende"[1] Operationen.

Eine Operation x_α wird als blockiert von y_β bezeichnet, wenn

1. $y_\beta < x_\alpha$
2. y_β blockiert ist und
3. die Arbeitsfolge $x_{(\beta<\alpha)}$ im Programm erscheint.

Folglich muß zu jeder blockierten Operation x_α mindestens eine andere blockierte Operation y_β gehören; z.B. sei a_1 von b_2 blockiert und es gelte

1. ... a_1 ...
2. ... b_2 ... a_2 ... mit $\overline{A}$.

Die Operation b_2 kann jetzt aber nicht von einer Arbeit aus Folge 1 blockiert werden, da dann zwischen 1. und 2. ein Zyklus mit $\overline{A}B$ bestünde. Das wiederum würde der Annahme über die Existenz der Zyklen widersprechen, daß nämlich die oben angegebenen Bedingungen für weniger als q + 1 Folgen zutreffen. Man muß also folgern, daß b_2 von c_3 blockiert

[1] Vgl. Akers, S.B.; Friedman, J.: A non-numerical ..., a.a.O., S. 437.

wird und erhält die Maschinenfolgen:

$$
\begin{array}{rl}
1. & \ldots\ a_1\ \ldots \\
2. & \ldots\ b_2\ \ldots\ a_2\ \ldots \\
3. & \ldots\ c_3\ \ldots\ b_2\ \ldots \\
 & \vdots \\
q\text{-}1. & \ldots\ x_{(q-1)}\ \ldots\ w_{(q-1)}\ \ldots \\
q. & \ldots\ y_q\ \ldots\ x_q\ \ldots
\end{array}
$$

mit dem Programm $A_{(2<1)}$, $B_{(3<2)}$, ..., $x_{(q<q-1)}$. Jetzt muß geprüft werden, ob y_q von a_1 blockiert wird und die entsprechende Bedingung $Y_{(q<1)}$ heißen kann. Da die Operation $Z_{(q+1)}$ voraussetzungsgemäß blockiert ist, muß es eine Operation u_γ geben, derart, daß z_{q+1} von u_γ blockiert wird: (sonst wird das Programm an dieser Stelle ja nicht blockiert)

$$
\begin{array}{rl}
q+1. & \ldots\ z_{q+1}\ \ldots \\
\gamma\ . & \ldots\ u_\gamma\ \ldots\ z
\end{array}
$$

mit $Z_{(\gamma<q+1)}$.

Ohne Einschränkung der Allgemeingültigkeit ergibt sich somit, daß y_q von z_{q+1} und z_{q+1} von a_1 blockiert wird:

$$
\begin{array}{rl}
1. & \ldots\ a_1\ \ldots\ z_1\ \ldots \\
2. & \ldots\ b_2\ \ldots\ a_2\ \ldots \\
3. & \ldots\ c_3\ \ldots\ b_3\ \ldots \\
 & \vdots \\
q. & \ldots\ y_q\ \ldots\ x_q\ \ldots \\
q+1. & \ldots\ z_{q+1}\ \ldots\ y_{q+1}\ \ldots
\end{array}
$$

mit $A_{(2<1)}$, $B_{(3<2)}$, ..., $Y_{(q+1<q)}$, $Z_{(1<q+1)}$.

Alle Programme, für welche sich derartige Zyklen nicht nachweisen lassen, sind realisierbar, allerdings ohne Berücksichtigung mehr oder weniger langer Verzögerungen, falls einzelne Operationen vorübergehend nicht ausgeführt werden können. Die Frage nach diesen Wartezeiten wird relevant bei der Auswahl der optimalen Programme.

2223. Kritik und Erweiterungen

Zunächst sind die Zusammenhänge an einem Beispiel zu verdeutlichen. Für die Arbeiten 1 und 2 seien die Bedingungen

$$1.\quad a_1 < b_1 < c_1 < d_1 < e_1$$

$$\text{und } 2.\quad e_2 < c_2 < a_2 < b_2 < d_2 \text{ gegeben.}$$

Bei den Maschinenpaaren ac, ae, bc, be, ce und de sind die Reihenfolgen der Arbeiten in 1. und 2. vertauscht, d.h. gemäß den Forderungen des Beweises sind alle Programme mit den Anordnungen

$$\overline{AC},\ \overline{AE},\ \overline{BC},\ \overline{BE},\ \overline{CE} \text{ und } \overline{DE}$$

nicht realisierbar.

Die $(21)^5 = 32$ theoretisch möglichen Zuordnungen (vgl. Tabelle 5)[1] sind also zu prüfen, ob sie durch die unverträglichen Forderungen blockiert werden.

$\overline{A}$	A	$\overline{A}$	A	$\overline{A}$	A	$\overline{A}$	A	$\overline{A}$	A	$\overline{A}$	A	$\overline{A}$	A	$\overline{A}$	A	$\overline{A}$	A	$\overline{A}$	A	$\overline{A}$	A	$\overline{A}$	A	$\overline{A}$	A	$\overline{A}$	A	$\overline{A}$	A	$\overline{A}$	A
$\overline{B}$	$\overline{B}$	B	B	$\overline{B}$	$\overline{B}$	B	B	$\overline{B}$	$\overline{B}$	B	B	$\overline{B}$	$\overline{B}$	B	B	$\overline{B}$	$\overline{B}$	B	B	$\overline{B}$	$\overline{B}$	B	B	$\overline{B}$	$\overline{B}$	B	B	$\overline{B}$	$\overline{B}$	B	B
$\overline{C}$	$\overline{C}$	$\overline{C}$	$\overline{C}$	C	C	C	C	$\overline{C}$	$\overline{C}$	$\overline{C}$	$\overline{C}$	C	C	C	C	$\overline{C}$	$\overline{C}$	$\overline{C}$	$\overline{C}$	C	C	C	C	$\overline{C}$	$\overline{C}$	$\overline{C}$	$\overline{C}$	C	C	C	C
$\overline{D}$	$\overline{D}$	$\overline{D}$	$\overline{D}$	$\overline{D}$	$\overline{D}$	$\overline{D}$	$\overline{D}$	D	D	D	D	D	D	D	D	$\overline{D}$	$\overline{D}$	$\overline{D}$	$\overline{D}$	$\overline{D}$	$\overline{D}$	$\overline{D}$	$\overline{D}$	D	D	D	D	D	D	D	D
$\overline{E}$	$\overline{E}$	$\overline{E}$	$\overline{E}$	$\overline{E}$	$\overline{E}$	$\overline{E}$	$\overline{E}$	$\overline{E}$	$\overline{E}$	$\overline{E}$	$\overline{E}$	$\overline{E}$	$\overline{E}$	$\overline{E}$	$\overline{E}$	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E

Tabelle 5

[1] Die Anordnung der Strukturregeln in Tabelle 5 ergibt sich wiederum durch binäre Addition bei 1 beginnend.

Da die unverträglichen Strukturregeln nicht in den Programmen erscheinen dürfen, werden sie als Bedingungen für deren Zulässigkeit negativ wie folgt formuliert: Nicht $\bar{A}C = (\overline{\bar{A}C})$, nicht $(\overline{\bar{A}E})$ usw. Da alle sechs Bedingungen erfüllt sein müssen (Konjunktivverknüpfung), entfällt jedes Programm, für welches schon eine Bedingung nicht erfüllt ist (d.h. die Forderung verneint wird); in Tabelle 4 erscheint dann an der entsprechenden Stelle eine Null, ist die Bedingung erfüllt, erscheint eine Eins. Für die aussagenlogische Funktion F, welche die Zulässigkeit fixiert, wird gefordert:

$$F = (\overline{\bar{A}C}) \cdot (\overline{\bar{A}E}) \cdot (\overline{\bar{B}C}) \cdot (\overline{\bar{B}E}) \cdot (\overline{\bar{C}E}) \cdot (\overline{\bar{D}E}) = 1!$$

| | | |

weisen Untersuchung der logischen Möglichkeiten jedes Programm aus, sobald eine der Bedingungen verneint wird[1]. Obwohl also von Schritt zu Schritt die Anzahl der notwendigen Operationen kleiner wird, sind diesem Vorgehen relativ enge Grenzen gesetzt. Das gilt auch im Hinblick auf die Entwicklung neuer leistungsfähiger Datenverarbeitungsanlagen.

Mit der Bestimmung der realisierbaren Programme ist das Lösungsverfahren für den allgemeinen Fall abgeschlossen. Lediglich für zwei Arbeiten und m Maschinen haben AKERS - FRIEDMAN zwei weitere Sätze formuliert, welche die Anzahl der realisierbaren und die optimalen Programme betreffen.

Die Anzahl A der realisierbaren Programme bei zwei Arbeiten und m Maschinen wird danach bestimmt als

$$A(m) = 1 + m + \#(\text{geordnete Paare}) + \#(\text{geordnete Tripel}) + \ldots + \#(\text{geordnete m-tupel})$$

Unter einem geordneten k-tupel ist dabei jede Teilmenge von k Maschinen zu verstehen, deren Elemente in beiden Maschinenfolgen in gleicher Weise angeordnet sind. (# = Anzahl).

Für das Beispiel ergibt sich:

$$A(5) = 11 = 1 + 5 + 4 \text{ (ab, ad, bd und cd als geordnete Paare)} + 1 \text{ (abd als geordnetes Tripel).}$$

[1] In unserem Beispiel also die Programme Nr. 5, 7, 13, 15, 21, 23, 29 und 31 bereits nach dem ersten Test.

Der Beweis dieses Satzes ist ausgehend vom Fall m = 1 durch vollständige Induktion über m + 1 für alle ganzen positiven Zahlen zu führen und leicht nachzuvollziehen. Darum kann hier auf eine Wiedergabe verzichtet werden[1].

Anders verhält es sich mit der Bedingung, durch welche die nicht möglich-optimalen Programme in einem zweiten Schritt aus der Menge der realisierbaren Zuordnungen eliminiert werden sollen. Wie sich zeigen wird, ist hier eine kritische Überprüfung notwendig.

Als Optimalitätskriterium wird für Ablaufplanungsmodelle regelmäßig die minimale Durchlaufzeit der Arbeitsobjekte angenommen. Das führt zu der Forderung, möglichst viele Arbeiten gleichzeitig auszuführen. Programme, deren Bearbeitungsvorschriften ständig oder zeitweilig nur die Durchführung einer Arbeit zur Zeit zulassen, können nicht optimal sein. Von zwei Arbeiten kann nun auf den beteiligten Maschinen aber immer nur dann bloß eine zur Zeit (d.h. allein) erledigt werden, wenn die andere Arbeit ihr auf eben derselben Maschine nachfolgt. In diesem Fall ist die Dauer von Arbeit 1 z.B. auf Maschine a Wartezeit für Arbeit 2, alle anderen m-1 Maschinen sind während dieser Zeitspanne und während der anschließenden Bearbeitungsdauer von Arbeit 2 auf Maschine a unbeschäftigt, sie stehen leer.

AKERS - FRIEDMAN definieren ein Aggregat, auf dem beide Arbeiten unmittelbar nacheinander ausgeführt werden müssen, während alle anderen Aggregate unbeschäftigt sind, als "freie" Maschine, und fordern für die Optimalität eines Programms, daß es keine freien Maschinen enthalten darf[2]. Da diese Auswahl ohne genaue Zeitzuordnungen erfolgt, muß sich das eigentlich-optimale Programm für eine bestimmte

---

[1] Vgl. Akers, S.B.; Friedman, J.: A non-numerical ..., a.a.O., S. 438.

[2] Vgl. Akers, S.B.; Friedman, J.: A non-numerical ..., a.a.O., S. 434 und 441.

Zeitzuordnung notwendig in der ermittelten Teilmenge befinden, bzw. es können Zeitzuordnungen angegeben werden, durch welche beliebige Programme aus der Teilmenge zu eigentlich-optimalen Programmen werden.

Nach AKERS - FRIEDMAN ist eine notwendige und hinreichende Bedingung, daß ein Programm keine "freie" Maschine enthält, darin gegeben, daß eine beliebige Maschine y hinsichtlich ihrer zeitlichen Anordnung innerhalb eines Programms folgende Beschränkungen erfüllt[1]:

(a) Für 1. $\ldots x_1 y_1 \ldots z_1 \ldots$

und 2. $\ldots x_2 \quad \ldots y_2 z_2 \ldots$

darf $X\overline{Y}Z$ nicht im Programm erscheinen.

Hat Maschinenfolge 1 die Form:

$$y_1 \ldots z_1 \ldots,$$

darf nicht $\overline{Y}Z$, oder Maschinenfolge 2 die Form:

$$\ldots x_2 \ldots y_2,$$

darf nicht $X\overline{Y}$ im Programm erscheinen.

(b) Für 1. $\ldots x_1 \quad \ldots y_1 z_1 \ldots$

und 2. $\ldots x_2 y_2 \ldots z_2 \quad \ldots$

darf nicht $\overline{X}Y\overline{Z}$ im Programm erscheinen.

Gilt für Maschinenfolge:

1. $\ldots x_1 \ldots y_1$

darf nicht $\overline{X}Y$ oder für 2. $y_2 \ldots z_2 \ldots$

darf $Y\overline{Z}$ nicht im Programm enthalten sein.

---

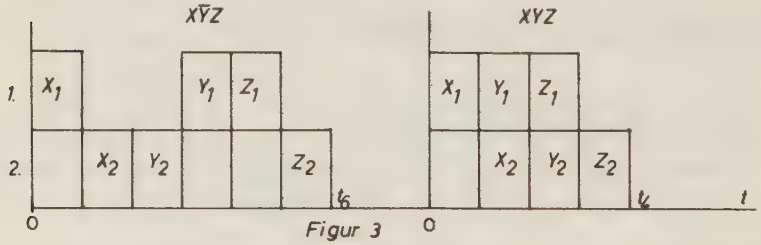
[1] Vgl. Akers, S.B.; Friedman, J.: A non-numerical ..., a.a.O., S. 441.

Daß diese Vorschriften zu "freien" Maschinen führen, ist leicht einzusehen, denn nach (a) gilt z.B. folgende zeitliche Anordnung:

$$x_1 < x_2 < y_2 < y_1 < z_1 < z_2$$

Da y_1 unmittelbar x_1 folgen soll (d.h. Arbeit 1 auf Maschine x direkt nach Maschine y entsprechend (a) 1.), y_1 aber erst nach Erledigung von y_2 in Angriff genommen werden kann (wegen $\overline{Y}$), andererseits unmittelbar nach y_2 Operation z_2 ausgeführt werden muß (gemäß (a) 2.), dabei aber $z_1 < z_2$ gilt, sind y_1 und y_2 allein auszuführen: Maschine y ist eine freie Maschine im Sinne der Definition; die Operationen y_1 und y_2 lassen eine simultane Durchführung anderer Arbeitsgänge nicht zu.

Entsprechend kann (b) nachgewiesen werden. Im GANTT-Diagramm[1] stellt sich dieser Zusammenhang wie folgt dar[2]:



Figur 3

Für den allgemeinen Beweis der Notwendigkeit der Bedingungen sei folgender Gedankengang angefügt: y_1 sei gemäß der Annahme von y als "freier" Maschine allein auszuführen. Sofern mit y_2 der Prozeß nicht beendet ist, ergibt sich, daß auf y_2 eine Operation z_2 folgt, für welche nur Z gelten kann.

---

[1] ZumGantt-Diagramm vgl. die ausführliche Darstellung bei Tempel, K.-H.: Wartezeitmodelle, Diss. Göttingen 1964, S.140 ff.

[2] Für die Gantt-Diagramme und die folgenden Ausführungen definieren wir jede Stillstandszeit (Leerzeit) eines Aggregates i mit l_i, die gesamte Leerzeit eines Programms mit L und unterstellen aus Gründen der Anschaulichkeit l_i = const. für i = 1, 2, ..., m.

Denn für $\overline{Z}$ könnte ja z_2 gleichzeitig mit y_1 erledigt werden und y wäre keine "freie" Maschine; das Programm mit $\overline{Y}$ muß also Z enthalten. Für die Kombination Z und $z_1 < y_1$ ergibt sich gleichfalls, daß y keine "freie" Maschine ist, da z_1 gleichzeitig mit y_2 erledigt werden könnte. Die weiteren theoretisch möglichen Kombinationen Z mit $z_2 < y_2$ bzw. $y_1 < z_1$ sind technisch nicht realisierbar (Zyklus).

Sind y weitere Aggregate vorgeschaltet, so kann für das letzte nur X gelten. Denn für $\overline{X}$ könnte y_2 simultan mit x_1 ausgeführt werden, bzw. für $x_1 < y_1$ und $y_2 < x_2$ ergäbe sich hinsichtlich y_2 und y_1 keine Direktfolge. Die Kombination X mit $x_1 < y_1$ und $y_2 < x_2$ ermöglicht die gleichzeitige Durchführung von x_1 und y_2 bzw. x_2 und y_1; X mit $y_1 < x_1$ oder $x_2 < y_2$ ist nicht realisierbar. Entsprechend können alle weiteren unzulässigen Anordnungen getestet werden.

Der Feststellung jedoch, daß die angegebenen Bedingungen hinreichend sind, können wir nicht zustimmen. Für die mathematische Richtigkeit der Argumentation muß die implikative Verknüpfung von Bedingung - Ereignis umkehrbar sein, d.h. in unserem Fall, sobald eine "freie" Maschine gegeben ist, muß eine der angeführten Bedingungen erfüllt sein. Dies trifft - wie zu zeigen sein wird - bei dem Satz von AKERS und FRIEDMAN nicht zu.

Für das obige Beispiel sind folgende elf Programme als realisierbar ermittelt worden:

$\overline{A}$ A $\overline{A}$ A A $\overline{A}$ A $\overline{A}$ A A A

$\overline{B}$ $\overline{B}$ B B B $\overline{B}$ $\overline{B}$ B B B B

$\overline{C}$ $\overline{C}$ $\overline{C}$ $\overline{C}$ C $\overline{C}$ $\overline{C}$ $\overline{C}$ $\overline{C}$ C C

$\overline{D}$ $\overline{D}$ $\overline{D}$ $\overline{D}$ $\overline{D}$ D D D D D D

$\overline{E}$ $\overline{E}$ $\overline{E}$ $\overline{E}$ $\overline{E}$ $\overline{E}$ $\overline{E}$ $\overline{E}$ $\overline{E}$ $\overline{E}$ E

Entsprechend den Sätzen (a) und (b) sind die Strukturregeln $\overline{A}BD$, $C\overline{D}$ und E unverträglich. Es ergibt sich somit für die elf Programme die Forderung

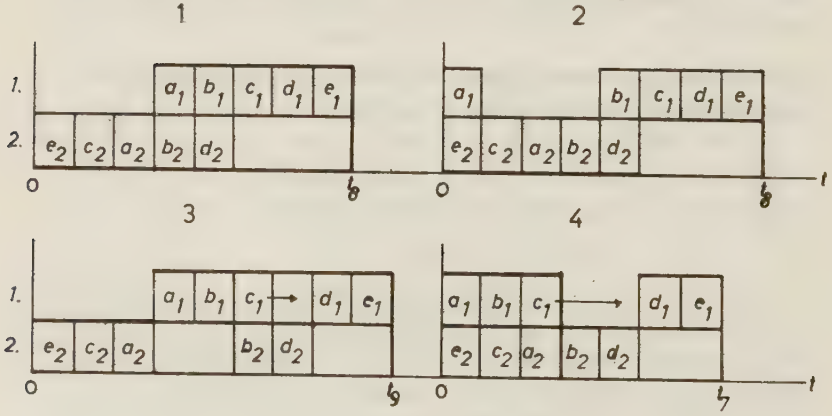
Nicht	$A\overline{B}D$ $(\overline{A\overline{B}D})$	1 1 1 1 1 1 0 1 1 1 1
	und	
nicht	$C\overline{D}$ $(\overline{C\overline{D}})$	1 1 1 1 0 1 1 1 1 1 1
	und	
nicht	E $(\overline{E})$	1 1 1 1 1 1 1 1 1 1 0
Funktionswert		1 1 1 1 0 1 0 1 1 1 0

Tabelle 8

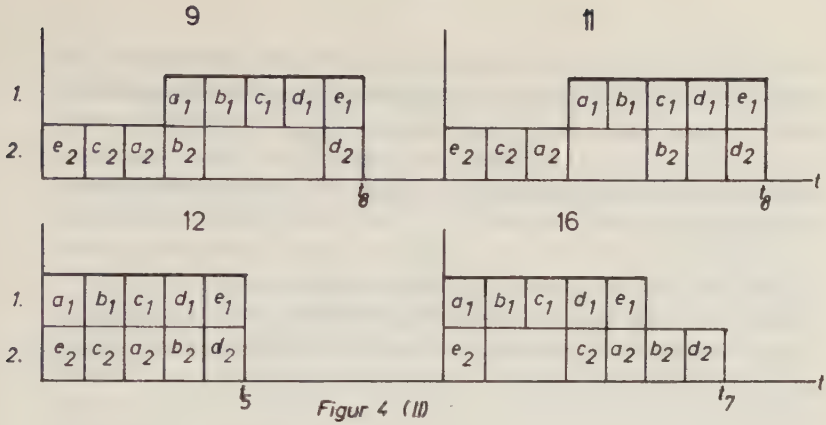
Die Anwendung der Vorschriften von AKERS - FRIEDMAN führt zu acht möglich optimalen Programmen, die - sollte die Hinlänglichkeit der Bedingung zutreffen - keine freien Maschinen mehr enthalten dürften.

Die GANTT-Diagramme zeigen jedoch, daß auch die Programme 3 und 11 "freie" Maschinen enthalten (jeweils Maschine a; d in Programm 3 und 4 ist dagegen nicht frei, da - wie mit Pfeil angedeutet - c_1 verschoben werden kann).

GANTT-Diagramme für möglich-optimale Programme



Figur 4 (1)



Figur 4 (II)

Optimale Programme im Sinne der Definition sind Nr. 1, 2, 4, 9, 12 und 16. Für l_i = constant ist Nr. 12 das gar-optimale Programm.

Wie hier für den speziellen Fall kann allgemein gezeigt werden, daß für die Hinlänglichkeit der Bedingungen ein weiterer Satz (c) angefügt werden muß.

Allgemein gilt, daß für die Maschinenfolgen

1. $a_1 \dots x_1y_1 \dots m_1$; bzw. $x_1y_1 \dots m_1$; bzw. $a_1 \dots x_1y_1$;

2. $a_2 \dots x_2y_2 \dots m_2$; $a_2 \dots x_2y_2 \dots m_2$; $a_2 \dots x_2y_2 \dots m_2$;

nicht $\overline{X}Y$,

für

1. $a_1 \dots x_1y_1 \dots m_1$; bzw. $a_1 \dots x_1y_1 \dots m_1$;

2. $x_2y_2 \dots m_2$; $a_1 \dots x_2y_2$;

nicht $X\overline{Y}$

und für

1. $x_1y_1 \dots m_1$; bzw. $a_1 \dots x_1y_1$; bzw. $x_1y_1 \dots m_1$; bzw. $a_1 \dots x_1y_1$;

2. $x_2y_2 \dots m_2$; $a_2 \dots x_2y_2$; $a_2 \dots x_2y_2$ $x_2y_2 \dots m_2$;

weder $\overline{X}Y$ noch $X\overline{Y}$ im Programm erscheinen dürfen.

Der vorliegende Modellansatz kann noch in einer anderen Hinsicht erweitert werden. Geht man für alle Arbeitsgänge von gleichen oder nahezu gleichen Operationszeiten aus (eine keineswegs unrealistische Annahme), so kann eine Bedingung für die Bestimmung der absolut optimalen Programme angegeben werden.

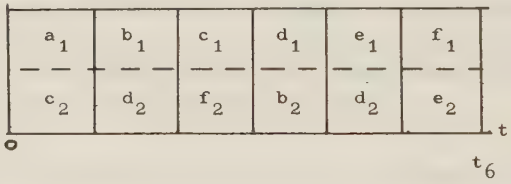
Optimal in diesem Sinn sind alle Programme, deren Prozeßregeln für alle die Maschinen "gestrichen" sind (z.B. $\overline{X}$), deren Reihenfolgen für beide Arbeiten vertauscht sind, die Leerzeiten sind in diesem Fall gleich Null.

Gilt z.B. für

$$1.\quad a_1 < b_1 < c_1 < d_1 < e_1 < l_1$$

$$2.\quad c_2 < d_2 < f_2 < b_2 < q_2 < e_2$$

die Regel A B $\overline{C}$ $\overline{D}$ E F, erhält man



Außerdem kann für Prozesse mit gleichen Maschinenfolgen für die Arbeiten und normaler Anordnung der Arbeiten (d.h. nur $x_{\underline{1}} < x_{\underline{2}}$) auf den Maschinen die Gesamtstillstandszeit L fixiert werden mit

$$L = (m^2 - m)\, l_i \qquad (i = 1,2,\ldots,m)$$

Das Verdienst von AKERS und FRIEDMAN ist ein zweifaches. Einmal liegt es darin, daß sie das bislang nur unvollkommen gelöste Ablaufplanungsproblem mit unkonventionellen Mitteln erfolgreich in Angriff genommen haben. Zweitens muß bemerkt werden, daß hier erstmals explizit eine konkrete Anwendungs-

möglichkeit der BOOLEschen Algebra für betriebliche Probleme realisiert worden ist. Insofern ist hier Pionierarbeit geleistet worden.

Selbstverständlich ist das entwickelte Verfahren kein systematischer Algorithmus, der in eindeutiger Weise zu einer Optimallösung führt; darüber hinaus ist er nur für kleinere Probleme operational. Da besondere numerische Werte nicht erforderlich sind, können jedoch die zahlreichen Ablaufplanungsprobleme, deren Zeitwerte veränderlich sind (z.B. im Baugewerbe infolge der Witterungseinflüsse oder bei mehr handwerklich-künstlerischer Tätigkeiten) kalkülartig behandelt werden.

Einer mathematischen Weiterentwicklung, die zu einer Ausdehnung der Sätze für die Bestimmung der optimalen Programme im allgemeinen Fall führen müßte, kann keine große Bedeutung beigemessen werden, denn aufgrund der theoretischen Konzeption des Modells werden alle Verfeinerungen auch für leistungsfähige Rechenanlagen kaum verwertbar sein. Nur rein theoretische Bedeutung haben auch die im Anschluß an AKERS und FRIEDMAN von GIFFLER - THOMPSON[1] und SZWARC[2] entwickelten Methoden. Beim ersten Ansatz handelt es sich praktisch um ein spezielles Probierverfahren, beim zweiten um ein graphisches Modell, dem naturgemäß von vornherein Grenzen hinsichtlich der Anzahl der zu berücksichtigenden Maschinen und Arbeiten gesetzt sind. Auch der Gedanke, komplexe Probleme in lösbare Teilaufgaben aufzuspalten, um über Suboptimierung zum Ziel zu kommen, hat kaum mehr als theoretische Bedeutung.

---

[1] Vgl. Giffler, B.; Thompson, G.L.: Scheduling General ..., a.a.O.

[2] Vgl. Szwarc, W.: Solution of the ..., a.a.O., s.a. Akers, S.B.: A Graphical Approach ..., a.a.O.

DRITTES KAPITEL

DAS SYSTEM DER BOOLESCHEN PROGRAMMIERUNG

31. Der Weg zu BOOLEschen Programmen in der Unternehmensforschung

Die im vorigen Kapitel behandelte aussagenlogische Programmierung bezieht sich - wie gezeigt wurde - auf den Wahrheitswert von Aussagen als logische Alternativen eines komplexen Entscheidungsproblems, die Entscheidungsvariablen selbst sind dabei als zweiwertig[1] vorherbestimmt - im Aussagenkalkül mit wahr oder falsch. Diese Einschränkung läßt vermuten, daß entsprechende Ansätze infolge der mangelnden Isomorphie[2] zwischen Problem- und Modellstruktur innerhalb der Unternehmensforschung nur selten formuliert werden können.

Andererseits jedoch ist eine Vielzahl praktischer Probleme dadurch gekennzeichnet, daß für sie nur ganzzahlige Lösungen sinnvoll sind: z.B. bei Fragen der Produktionsplanung mit unteilbaren Potentialfaktoren[3]. Entsprechend der Häufigkeit dieser Probleme wurde eine Reihe von speziellen Algorithmen entwickelt[4]; am bekanntesten ist das Verfahren von

---

[1] Man kennt auch mehrwertige logische Systeme, z.B. dreiwertige, da aber "mit einer andern Basis als 2 auch nicht mehr beschrieben werden kann - ... wird tatsächlich nur der Formalismus unübersichtlicher."
Zemanek, H.: Logische Algebra und Theorie der Schaltnetzwerke, in: Taschenbuch der Nachrichtenverarbeitung, Hrsg. Karl Steinbuch Berlin - Göttingen - Heidelberg 1962, S.102.

[2] Vgl. Kosiol, E.: Modellanalyse als Grundlage unternehmerischer Entscheidungen, in ZfhF 13. Jg. N.F. 1961, S. 321.

[3] Vgl. Dantzig, George B.: Discrete-variable Extremum Problems, in: JORSA Vol. 5 1957 No. 2 S. 266 ff., der eine Zusammenstellung ganzzahliger Programme bringt; s.a. Krelle, W.: Ganzzahlige Programmierungen. Theorie und Anwendungen in der Praxis, in: Ufo (Operations Research) Bd.2 1958 S. 161 ff.

[4] Vgl. Balinski, M.L.: Integer Programming: Methods, Uses, Computation, in: M.S. Vol 12 1965 No. 3, S. 253 ff., welcher alle bekannten Verfahren darstellt.

RALPH E. GOMORY , welches auf dem bekannten Simplex-Kriterium aufbaut[1].

Eine besondere Form der Ganzzahligkeit ist offenbar dann gegeben, wenn die Variablen nur die Werte 0 oder 1 annehmen können, in diesem Fall liegt eine "außerordentliche Ganzzahligkeit"[2] vor.

DANTZIG bezeichnet ein Programm mit der Form

$$cx' = z$$

$$Ax' = b$$

$$\text{mit } x_j = 0 \text{ oder } 1$$

sogar als "the standard discrete form"[3].

Tatsächlich lassen sich nämlich eine Reihe aktueller Probleme von vornherein entsprechend formulieren:

z.B. Ernennungs- oder Zuordnungsprobleme[4]
(Assignment Problems) einschließlich
des sogenannten Marriage Problems[5]

---

[1] Vgl. Gomory, R.E.: Outline of an Algorithm for Integer Solutions to Linear Programs, in Bulletin of the American Mathematical Society, Vol. 64 1958 No. 5 S. 275 ff. s.a.
Angermann, A.: Entscheidungsmodelle ..., a.a.O., S. 216 ff.

[2] Eichenberger, Heinrich: Ökonomische Methoden in der linearen Planungsrechnung, Hamburg - Berlin 1963 S. 95.

[3] Dantzig, G.B.: Discrete-Variable ..., a.a.O., S. 268. Hier ist c als Zeilen-, x' als Spaltenvektor zu verstehen, während z als Ausdruck für die Zielfunktion Maximal- oder Minimalwertbestimmung bedeutet.

[4] Vgl. Eichenberger, H.: Ökonomische Methoden ..., a.a.O., S. 92 ff.
Krelle, W.: Ganzzahlige ..., a.a.O., S. 165.
Dantzig, G.B.: Lineare Programmierung und Erweiterungen, Berlin - Heidelberg - New York 1966.

[5] Vgl. Dantzig, G.B.: Diskrete-Variable ..., a.a.O., S. 267.

Lokalisationsprobleme[1]
Reisenden-(Traveling-Salesman)probleme[2]
Das Rucksack-(Knapsack)problem[3]
Auslieferungs-(Delivery)-probleme[4]
Netzwerk-Fluß-Probleme[5] u.a.

Darüber hinaus ist es in vielen Fällen möglich, ein ganzzahliges Programm derart umzuformen, daß die Variablen nur noch die Werte 0 oder 1 annehmen können[6]. Diese Umformung ist immer möglich, wenn für die ursprünglichen Variablen eine obere ganzzahlige Grenze (m_j) angegeben werden kann[7].

Für ganzzahlige x_j mit $0 \leq x_j \leq m_j$ ($j = 1, \ldots, n$) werden dann die neuen Variablen y_{ij} definiert, so daß x_j durch $x_{ji} + y_{j2} + y_{j3} + \ldots + y_{jmj}$ ersetzt wird[8].

---

[1] Vgl. Berg, R.: Planung von Büroräumlichkeiten, in: Industrielle Organisation Heft 9, 1957, S. 337.
Manne, Alan, S.: Plant Location under Economics of Scale Dezentralization and Computation, in: MS, Vol. 11, 1964, S. 213 ff.
Balinski, M.L.: Integer Programming ..., a.a.O., S. 286 ff.

[2] Vgl. Dantzig, G.B.; Fulkerson, D.R. and Johnson, S.: Solution of a large Scale Traveling Salesman Problem, in: JORSA 1954, Vol. 2, No. 4, S. 393 ff. besonders auch Flood, M.M.: The Traveling Salesman Problem, in: JORSA 1956, Vol. 4, No. 1, S. 61 ff., der auf die Ähnlichkeit zum Assignment-Problem hinweist.

[3] Vgl. Dantzig, G.B.: Discrete-Variable ..., a.a.O., S. 273 ff.

[4] Vgl. Balinski, M.L. and Quandt, R.E.: On an Integer Program for a Delivery Problem, in: JORSA 1964, Vol., 12, No. 2, S. 300 ff.

[5] Vgl. Dantzig, G.B.: Discrete-Variable ..., a.a.O., S. 270, und die im Abschnitt 433. dieser Arbeit angegebene Spezialliteratur zur FORD - FULKERSON-Theorie.

[6] Vgl. Krelle, W.: Ganzzahlige Programmierungen ..., a.a.O., S. 172.

[7] "There exists, however a very simple device by which such problems can be reduced to the 0-or-1 form if the variables have known upper bounds", in: Dantzig, G.B.: Discrete-Variable ..., a.a.O., S. 268.

[8] Vgl. außer Krelle, W.: Ganzzahlige Programmierungen ..., a.a.O., S. 268, auch
Dantzig, G.B.: On the Significance of Solving Linear Programming Problems with some Integer variables, in: Econometrica 1960, Vol. 28 No. 1, S. 30 ff., bes.S.35.

Geht man von der für praktische betriebliche Probleme regelmäßig zutreffenden Annahme aus, daß für die wirtschaftlichen Aktivitäten des Entscheidungskalküls obere und untere Grenzen gegeben sind (z.B. technische oder marktliche Kapazitäten), dann kann allgemein für jeden Ausdruck f_i von den Variablen x_1 bis x_n die Binärdarstellung wie folgt entwickelt werden[1].

Es gelte

$$f_i\ (x_1, x_2, \ldots, x_n) \quad (i = 1, \ldots, m)$$

für die Variablen x_j $(j = 1, \ldots, n)$ mit den Grenzen $\underline{m}_j \leq x_j \leq \overline{m}_j$.

Die Differenz $x_j - \underline{m}_j$ wird mit y_j, die Differenz $\overline{m}_j - \underline{m}_j$ mit M_j bezeichnet. $(M_j \geq y_j \geq 0)$.

Die Differenz y_j wird gemäß der üblichen Binärentwicklung[2] durch die neuen Variablen wie folgt ausgefüllt:

$$y_j = 2^0 y_{0j} + 2^1 y_{1j} + \ldots + 2^{r(j)} y_{r(j)j}$$

$$y_{kj} = \text{boolesch!}$$

Die Anzahl der neuen Variablen hängt naturgemäß von der Intervallgröße M_j ab, aus $2^{r(j)} \leqq M_j$ folgt

$$r(j) \leqq \log_2 M_j$$

---

[1] Vgl. hierzu Ivanescu, Petru L.: Programmation polynomiale en nombres entiers, in: Comptes Rendus Hebdomadaires Des Seances de l'Academie Des Sciences Jg. 1964, Gr. 1, S. 424 ff.

[2] Vgl. Sommer, F.: Einführung ..., a.a.O., S. 3 f.

Mit der Gleichung für die Differenz y_j ergibt sich aus

$p_i(x_1,\ldots,x_n)$ bzw.

$$p_i(y_1 \pm \underline{m}_1, y_2 \pm \underline{m}_2, \ldots, y_n \pm \underline{m}_n)$$

$$p_i(2^0 y_{01} + 2^1 y_{11} + \ldots + 2^{r(1)} y_{r(1)} \pm \underline{m}_1, 2^0 y_{02} + 2^1 y_{12} +$$
$$+ \ldots + 2^{r(2)} y_{r(2)2} \pm \underline{m}_2, \ldots, 2^0 y_{0n} + 2^1 y_{1n} + \ldots + 2^{(n)} y_{(n)n} \pm \underline{m}_n)$$

Durch entsprechende Aufspaltung der Aktivitäten können eine Reihe wirtschaftlicher Probleme in einen zweiwertigen Ansatz gebracht werden. Als Beispiel für ein derartiges Vorgehen sei das Verfahren von MARKOWITZ - MANNE erwähnt[1].

Dabei muß jedoch beachtet werden, daß durch diese Umformung die Programme auch für die mechanische Durchrechnung zu unhandlich werden können, zumal die Konvergenz der herkömmlichen Algorithmen für die ganzzahlige Programmierung relativ schlecht ist[2]. Besonders "für diese spezielle ganzzahlige Programmierung (with zero-one variables; with Boolean variables), für die zahlreiche Anwendungsmöglichkeiten bestehen, sind die erwähnten Verfahren weniger geeignet", stellt DINKELBACH fest[3]. Daran ändert auch der Umstand

---

[1] Vgl. Markowitz; Harry, M. and Manne, Alan S.: On the Solution of Discrete Programming Problems in: Econometrica, Vol. 25, 1957, S. 84 ff.

[2] Vgl. hierzu Albach, H.: Maschinenbelegungspläne bei Einzelfertigung, Sonderdruck des Landesamtes für Forschung des Landes Nordrhein-Westfalen, Köln und Opladen 1965, S. 23, welcher bei der Ermittlung optimaler Maschinenbelegungspläne mit dem Verfahren von GOMORY keine befriedigenden Erfahrungen gemacht hat.
Balenski, M.L.: Integer Programming ..., a.a.O., S.253 ff., zu einzelnen Verfahren bes. S. 294 ff.
Balenski, M.L. and Quandt, R.E.: On an Integer ..., a.a.O., S. 302 f.
S. auch Krelle, W.: Gelöste und ungelöste Probleme der Unternehmensforschung, in: Sitzungsbericht vom 4.10.1961 der Arbeitsgemeinschaft für Forschung des Landes Nordrhein-Westfalen, Köln und Opladen 1961, S. 21 f.

[3] Dinkelbach, W.: Zum Problem der Produktionsplanung in Ein- und Mehrproduktunternehmen, Würzburg - Wien 1964, S. 15.

nichts, daß es manchmal möglich ist, durch besondere Operationen die Anzahl der Variablen vorab wieder zu vermindern. Die Kenntnis dieser Schwierigkeiten einerseits und die Tatsache, daß viele praktische Probleme in zweiwertige Ansätze gebracht werden können auf der anderen Seite haben zu der Frage geführt, ob man nicht mit speziellen , der zweiwertigen Problemstruktur von vornherein adäquaten algebraischen Mitteln (Isomorphie des Modells!) weiterkommt. Als Grundlage hierfür bietet sich das System der BOOLEschen Algebra an.

Die ersten und grundlegenden Untersuchungen in dieser Richtung hat der französische Mathematiker R. FORTET 1959/60 veröffentlicht[1]. Hierzu soll bereits jetzt festgestellt werden, daß die BOOLEsche Algebra für die operationsanalytische Behandlung ganzzahliger Ansätze - fortan BOOLEsche Programmierung genannt - bislang nur für Probleme möglich ist, die nicht nur eo ipso eine zweiwertige Struktur haben, sondern darüber hinaus den logischen Gesetzen der BOOLEschen Algebra genügen. Denn bei der Reduzierung anderer Probleme auf BOOLEsche Programme tritt die zusätzliche Schwierigkeit auf, außer den Variablen auch die zwischen ihnen bestehenden arithmetischen Verknüpfungen entsprechend zu transformieren. Für derartige Umformungen aber bemerkt FORTET, wüßte er kein allgemeines Verfahren[2]. Bekannt ist uns hier der Versuch von IVANESCU[3], der die Polynome des ursprüngli-

---

[1] Vgl. Fortet, R.: L'Algebre de Boole et ses Applications en Recherche Operationnelle, in: Cahiers Du Centre D'Etudes De Recherche Operationnelle 1959 No. 4, S. 5 ff. derselbe: Applications de l'Algebre de Boole en Recherche Operationnelle, in: Revue Francaise de Recherche Operationelle, 4. Jg. 1960 No. 14, S. 17.

[2] Vgl. Fortet, R.: L'Algebre de Boole ..., a.a.O., S. 13: "Je ne sais pas d'une facon generale quelle est la classe des equations algebrigues entiers qui se laissent resoudre pour un procede simple systematique."

[3] Vgl. Ivanescu, P.L.: Programmation polynomiale ..., a.a.O., S. 424 ff.

chen Programms nach einem Vorschlag von CAMION[1] in ein System von Kongruenzen[2] überführt und dann auf der Basis dieses endlichen kommutativen Körpers eine äquivalente Lösung fixiert. Dieser Ansatz hat nur theoretische Bedeutung, auf das Lösungsverfahren selbst werden wir später noch genauer eingehen.

32. Die Grundstruktur BOOLEscher Programme

Im vorigen Abschnitt ist eine Reihe von Problemen aufgezählt worden, welche eine zweiwertige Problemstruktur aufweisen und somit grundsätzlich BOOLEschen Ansätzen zugänglich sind. Hier soll - bevor auf die Technik der BOOLEschen Programmierung eigegangen wird - an einem konkreten Beispiel, dem Auswahl-(Selection)problem das typische BOOLEsche Programm erläutert werden.

Ein Hauptmerkmal der neueren gegenüber der klassischen Produktions- und Kostentheorie ist die Prämisse von unteilbaren, jedoch polyvalenten Produktionsfaktoren und damit verbunden der Übergang von peripheren zu alternativen Substitutions-

---

[1] Vgl. Camion, P.: Une Methode de resolution par l'Algebre de Boole des problemes combinatoires ou interviennent des entiers, in: Cahiers Du Centre D'Études de Recherche Operationnelle, Vol. 2, 1960, No. 3, S. 234 ff.

[2] Ganze Zahlen heißen kongruent modulo m, wenn sie bei ihrer Division durch m den gleichen Rest ergeben. Bei Division durch m = 2 ergibt sich stets als Rest 0 oder 1, Kongruenzen mod 2 sind damit als sog. Restklassenring durch das 0, 1 System fixiert, für welches entsprechende additive und multiplikative Verknüpfungen definiert werden können. Vgl. hierzu Courant, R. und Robbins, H.: Was ist Mathematik?, a.a.O., S. 26 ff.
Van der Waerden, B.L.: Algebra, 1. Teil, Berlin - Göttingen - Heidelberg 1960, S. 128 ff.

prozessen[1]. ANGERMANN ordnet die in einem Betrieb vorkommenden alternativen Substitutionsentscheidungen zwei typischen Fällen zu: Diejenigen Probleme, bei denen einige von den zur Wahl stehenden alternativen Faktoren zur Erreichung eines vorgegebenen Ziels bestimmt werden müssen, erfordern selektive Substitutionsentscheidungen, die entsprechenden Entscheidungsmodelle werden als Selektionsmodelle bezeichnet[2]. Müssen dagegen mehrere bereits vorhandene Produktionsfaktoren bestimmten betrieblichen Aufgaben zugewiesen werden, sind assignative Substitutionsentscheidungen zu treffen. Sie deckt das oben erwähnte Assignationsmodell. Nur wenn die Zahl der möglichen Alternativen begrenzt ist, kommt man durch einfachen rechnerischen Vergleich der Auswahlkriterien bei verschiedenen Entscheidungen zum Ziel (Verfahrensvergleich), anderenfalls müssen spezielle Verfahren angewandt werden.

Ein Auswahlproblem besteht etwa darin, daß ein Unternehmen die Durchführung einer bestimmten Anzahl von Arbeitsoperationen plant. Der Investitionsgütermarkt bietet dazu viele unterschiedliche (in Preis, Größe, Technik usw.) Maschinen an: Es ist die Auswahl (das Investitionsprogramm) zu fixieren, welche bei minimalem Aufwand die Ausführung aller Arbeiten ermöglicht. Die Entscheidung bezüglich der einzelnen Maschine kann nur ja oder nein, 1 oder 0 lauten, d.h. sie

---

[1] In der neueren Produktionstheorie wird denn auch häufig mit Variablen gerechnet, die nur 0 oder 1 sein können, wobei die Beziehung zur BOOLEschen Algebra nicht oder nur am Rande erwähnt wird.
Vgl. z.B. Dinkelbach, W.: Zum Problem der Produktionsplanung in Ein- und Mehrproduktunternehmen, Würzburg - Wien 1964, S. 22 und die dort angegebenen Literaturstellen.
Riester, W.F.: Planungsmodelle im Steinkohlenbergbau, in: Gegenwartsfragen der Unternehmensführung, Festschrift zum 65. Geburtstag von W. Hasenack, Herne-Berlin 1966, S. 226.

[2] Vgl. Angermann, A.: Entscheidungsmodelle, a.a.O., S. 117.

bleibt in der Auswahl oder nicht. Entsprechendes gilt z.B. für die einzelne Teilstrecke eines komplexen Rundreisestreckensystems[1] oder für bestimmte Knotenpunkte eines Verkehrsnetzes bzw. Standorte eines neu aufzubauenden Niederlassungssystems usw.: Entscheidung ist hier die Alternation, graduelle Abstufungen können nicht berücksichtigt werden. Es handelt sich also bei dieser Art von Entscheidungen um ein klares Entweder - Oder; fällt die Entscheidung für eine Alternative, entfällt die andere Möglichkeit. Daß diese Modellansätze nicht nur auf Probleme mit unteilbaren Produktionsfaktoren beschränkt sind, zeigen folgende Situationen: Im Einzugsgebiet einer Erdölraffinerie liegen zahlreiche Oelvorkommen, sie unterscheiden sich sowohl in ihrer Ergiebigkeit als auch in der Entfernung zur Raffinerie. Geplant ist die Erschließung einiger Vorkommen, diese sollen durch Rohrleitungen mit der Verarbeitungsstelle verbunden werden. Welche Vorkommen sind auszuwählen. Auch die Projektauswahl bei Betrieben der Auftragsfertigung und das bekannte Schaufensterproblem[2] können entsprechend formuliert werden.

Diese Hinweise mögen genügen, um zu zeigen, daß es sich hier um eine spezielle Art betrieblicher Entscheidungssituationen handelt, deren Lösungsbedingungen hinsichtlich der Variablen BOOLEsche Zahlen fordern. Sie zeigen gleichzeitig die Bedeutung, die den Bestrebungen zukommt, hierfür geeignete Lösungsverfahren zu entwickeln. Die Beschränkung auf die bekannten Verfahren der ganzzahligen Programmierung erscheint als unzureichender Behelf. Im folgenden sind darum die Grundlagen der BOOLEschen und PSEUDO-BOOLEschen Ansätze theoretisch und praktisch darzustellen.

---

[1] "Da es sich bei dem Travelling Salesman Problem um eine 'Ja-Nein'-Alternativentscheidung handelt.,."
Ahlsdorff, M.: Probleme des innerbetrieblichen Transportes in betriebstheoretischer Sicht, Diss. Göttingen 1965, S. 209.

[2] Vgl. Angermann, A.: Entscheidungsmodelle..., a.a.O., S. 159 ff.

33. BOOLEsche und PSEUDO-BOOLEsche Ansätze

Im ersten Kapitel sind die algebraischen Grundlagen der Aussagenlogik behandelt worden. Es wurde gezeigt, daß die gleichen Axiome das mathematische Gerüst der BOOLEschen Algebra bilden. Jede Menge A von Elementen, für welche in bezug auf zwei bestimmte Elemente die binären Operationen der Multiplikation und Addition kommutativ, assoziativ und distributiv sind, und wenn für jedes Element a in A ein Element $\bar{a}$ (Negation) existiert, ist eine BOOLEsche Algebra[1]. BOOLEsche Multiplikation und Addition entsprechen der logischen Konjunktion und Disjunktion. Hier interessiert speziell die BOOLEsche Algebra, welche nur aus den beiden Elementen 0 und 1 besteht, für sie gilt:

+	0	1
0	0	1
1	1	1

·	0	1
0	0	0
1	0	1

a	0	1
$\bar{a}$	1	0

Die Aussagenvariablen der Aussagenlogik sind nunmehr als BOOLEsche Variable, die Aussagenfunktionen als BOOLEsche Funktionen definiert. Eine BOOLEsche Funktion ist danach jede Funktion f von n BOOLEschen Variablen $x_1,\ldots,x_n$, "welche keine anderen Werte als 0 oder 1 annehmen kann, d.h. selber eine BOOLEsche Zahl ist."[2]

---

[1] Vgl. ausführliche Definitionen bei Dwinger, Ph.: Introduction to Boolean Algebras, Würzburg 1961, bes. S. 5
Sikorski, Roman: Boolean Algebras, 2. Aufl. Berlin - Göttingen - Heidelberg - New York 1964 und
Whitesitt, J. Eldon: Boolesche Algebra und ihre Anwendungen, Braunschweig 1964,

[2] Fortet, R.: Applications de l'Algebre ..., a.a.O., S. 19: "qui ne peut prendre que les valeurs 0 ou 1, c'est-a-dire qui est elle-meme un nombre boolean."

Eine Begründung der BOOLEschen Programmierung auf diese Systeme allein würde bedeuten, daß sie von vornherein auf Probleme beschränkt wäre, für die nicht nur die Variablen, sondern auch die Beschränkungsgleichungen und die Zielfunktionen boolesch sein müßten. Zwar können - wie gezeigt wird - einige ganzzahlige Programme entsprechend formuliert werden, doch erscheint uns diese Auffassung zu eng. An dieser Stelle sollen die BOOLEschen Ansätze als Entscheidungsmodelle für alle betrieblichen Probleme entwickelt werden, die alternative Entscheidungen erfordern, denn die angeführten Beispiele machen deutlich, daß für "a great variety of problems in operations research and related areas, may be given a unified approach by means of mathematical programming with bivalent variables."[1] Wir beziehen in diese Untersuchung darum die Fälle mit ein, in denen nur alle Variablen der Funktionen boolesch sind, der Funktionswert jedoch beliebig ist. Derartige Funktionen entstehen durch die Abbildung eines BOOLEschen Ausdrucks in die Menge der reellen Zahlen. IVANESCU nennt sie "PSEUDO-BOOLEsche" Funktionen[2]. FORTET sieht die unmittelbare Beziehung zur ganzzahligen Programmierung und bezeichnet eine Funktion, deren Argumente boolesch sind, deren Koeffizienten jedoch beliebige reelle Werte besitzen als "fonction algebrique entiere"[3]. Beide Systeme sind identisch. Wichtig ist vor allem, daß wir zu den PSEUDO-BOOLEschen Ansätzen auch Verfahren rechnen, in denen neben logischen Verknüpfungen herkömmliche arithmetische Operationen vorkommen.

---

[1] Ivanescu, P.L.: Pseudo-Boolean Programming and Applications, erschienen als Lecture Note in Mathematics No. 9, Berlin - Heidelberg - New York 1965, S. 1.

[2] Vgl. Ivanescu, P.L.: Systems of Pseudo-Boolean Equations and Inequalities, in: Bulletin De L'Academie Polonaise des Sciences, Vol. XII. 1964, S. 673 ff.

[3] Vgl. Fortet, R.: Applications de L'Algebre ..., a.a.O., S.20 f. derselbe: L'Algebre de Boole ..., a.a.O., S. 12 ff.

Dieser Unterschied zu den BOOLEschen Ansätzen ist genau zu beachten. Werden Probleme untersucht, für welche nur ganze Werte relevant sind, kann die PSEUDO-BOOLEsche Funktion durch Abbildung in die Menge $\mathbb{Z}$ der ganzen Zahlen entwickelt werden[1]. Das ist jedoch nicht notwendig, um von PSEUDO-BOOLEschen Programmen sprechen zu können, es genügt, wenn die Variablen allein als boolesch vorbestimmt sind. Dieses Merkmal nämlich entscheidet über die Anwendungsmöglichkeit spezieller Lösungsverfahren.

Es ist somit notwendig, zwischen BOOLEschen und PSEUDO-BOOLEschen Programmen zu unterscheiden. Zunächst ist die BOOLEsche Programmierung zu behandeln.

34. BOOLEsche Programmierung

341. Mathematische Grundlegung

Die BOOLEsche Programmierung, wie sie hier dargestellt wird, ist eine konsequente Weiterentwicklung der aussagenlogischen Programmierung. Das wird deutlich, wenn die notwendigen mathematischen Grundlagen näher untersucht werden. Das Vorgehen der aussagenlogischen Programmierung besteht darin, komplexe Wirkzusammenhänge beliebiger Art in eine Symbolsprache zu übertragen und nach den Vorschriften des logischen Aussagenkalküls so umzuformen, daß alle logischen Abhängigkeiten so kurz und klar wie möglich erkennbar werden.

Das Hauptproblem innerhalb der BOOLEschen Programmierung, welches sich als "seltsam widerspenstig erwiesen hat", ist "the problem of devising a general mechanical procedure for reducing any formula to its simplest equivalent"[2]. Es mußte

---

[1] Darauf beschränkt sich IVANESCU in: Pseudo-Boolean Programming,.., a.a.O., S. 5 ff.

[2] Quine, W.V.: The Problem of Simplifying ..., a.a.O., S. 521.

also ein Verfahren entwickelt werden, durch welches systematisch jeder BOOLEsche Ausdruck in seine einfachste Form gebracht werden konnte. Bekannt ist dieses Problem als "the minimum disjunktive normal form problem"[1] oder allgemein als das Problem der Minimierung BOOLEscher Funktionen[2]. Inwiefern auch hier eine Beziehung zur herkömmlichen ganzzahligen Programmierung gegeben ist, zeigen folgende Überlegungen:

(1) Jeder BOOLEsche Ausdruck kann in eine Normalform gebracht werden. Bei der Behandlung der aussagenlogischen Programmierung sind die disjunktive und die konjunktive Normalform dargestellt worden. Hier wird für die Normalform zusätzlich gefordert, daß die disjunktiv oder konjunktiv verbundenen Glieder eines BOOLEschen Ausdrucks keine Variable zweimal enthalten und kein Glied doppelt vorkommt[3].

Eine in Normalform gebrachte BOOLEsche Funktion (BF) heißt entwickelt, wenn in allen ihren Gliedern (bei konjunktiver Verknüpfung Produkte genannt) jede Variable einmal (positiv oder negativ) vorkommt. BF können stets entwickelt werden, sie ändern ihren Wert dadurch nicht. Dieser Wert ist - wie bereits bei der aussagenlogischen Programmierung gezeigt - bei der vollständig entwickelten disjunktiven Normalform gleich 1. Jeder BF

---

[1] Balinski, M.L.: Integer Programming ..., a.a.O., S. 274, 282, s.a. S. 283: Das Problem ist to "find a 'cheapest' or 'simplest' expression or function in the x_j" bzw. "to find a solution containing the fewest number of products, or the solution which contains the fewest number of terms."

[2] z.B. bei Cobham, A.; Fridshal, R.; North, J.H.: An Application of Linear Programming to the Minimization of Boolean Functions, IBM-Research-Report RC - 472 1961 und
Cobham, A.; North, J.H.: Extensions of the Integer Programming Approach to the Minimization of Boolean Functions, IBM-Research-Report RC - 915 1963.

[3] Vgl. Quine , W.V.: The Problem ..., a.a.O., S. 521.

$f = f(x_1, \ldots, x_n)$ können 2^n BOOLEsche Vektoren zugeordnet werden, z.B. für $f = f(x_1, x_2, x_3)$

$$f = x_1x_2x_3 + \bar{x}_1x_2x_3 + x_1\bar{x}_2x_3 + \bar{x}_1\bar{x}_2x_3 + x_1x_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3 + x_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2\bar{x}_3$$

Diese bilden die Menge V(n):

$$V(n) = \{(1,1,1),(0,1,1),(1,0,1),(0,0,1),(1,1,0),(0,1,0),(1,0,0),(0,0,0)\}$$

oder $V(n) = \{v_1, \ldots, v_8\}$

Wird nun in einer BF jeder Variablen in dieser eindeutigen Weise der Wert 0 oder 1 zugeordnet, dann wird genau ein Glied gleich 1, alle anderen 0[1].

(2) Bezeichnet v_0 einen bestimmten Vektor aus der Menge V(n), so heißt der Ausdruck $p_{v_0}(v)$, der 1 wird, wenn $v = v_0$ und 0 in allen anderen Fällen, fundamentales oder vollständiges Produkt[2]. Z.B. ist für $v_0 = (0, 1, 0, 1)$ das Produkt

$$p_{v_0}(v) = \bar{X}_1X_2\bar{X}_3X_4$$

vollständig. Allgemein gilt:

$$p_{v_0}(v) = \bigwedge_j Y_j$$

wobei $Y_j = X_j$, wenn $x_j = 1$ und $Y_j = \bar{X}_j$, wenn $x_j = 0$.

Diese Formulierung ist als kanonischer Ausdruck des Produktes definiert. Entsprechend wird der kanonische Aus-

---

[1] Vgl. Fortet, R.: L'Algebre de Boole ..., a.a.O , S. 8.
[2] Vgl. Whitesitt, J.E.: Boolesche Algebra ..., a.a.O., S. 38.

druck für die ganze BF f(v) gebildet, der eine disjunktive Verknüpfung aller Glieder ist, welche für eine bestimmte Menge M ($M \subset V(n)$) aller Vektoren f(v) = 1 bestimmen, d.h.[1]

$$f(v) = \bigvee_{i \in M} p_i(v)$$

Die kanonische Zerlegung BF ist für die programmierte Bearbeitung BOOLEscher Programme besonders wichtig. Sie bildet den Ausgangspunkt für die systematische Vereinfachung der Funktionen. Allgemein kann jede BF in eine Form gebracht werden, welche durch die "systematische Anschreibung der Funktion als Disjunktion von Konjunktionen aller Variabler x_i"[2] entsteht.

$$f(x_1,\ldots,x_n) = \bigvee_{j}^{1,m} \quad \overset{1,n}{/\!/}_{i} \; x_i$$

Hierin gibt m die Anzahl der Disjunktionsglieder, d.h. der Vollkonjunktionen aller n Variablen an und ist gleich der Zahl möglicher Zuordnungen[3]; die vollständige (entwickelte) disjunktive Normalform einer BF von n Variablen hat 2^n Glieder.

---

[1] Das Zeichen $/\!/$ symbolisiert die Mehrfachkonjunktion beliebig vieler ariabler. Konjunktionen aller Variabler eines Ausdrucks werden als Minterme oder Vollkonjunktionen bezeichnet, entsprechend die Mehrfachdisjunktionen als Maxterme und durch $\bigvee$ oder $\sum$ dargestellt, wobei der Punkt andeutet, daß es sich um die Addition in einer BOOLEschen Algebra handelt.

[2] Zemanek, H.: Logische Algebra und Theorie der Schaltnetzwerke, in: Taschenbuch der Nachrichtenverarbeitung, a.a.O., S. 108.

[3] Vgl. hierzu Rosenberg, I.; Ivanescu, P.; Rudeanu, S.: An Application of Discrete Linear Programming of Boolean Functions (in Russisch), in: Revue De Mathematiques Pures et Appliques, Vol. III. 1963, No. 3, S. 465 f.

(3) Auf der Suche nach dem einfachsten Ausdruck einer BF in der disjunktiven Normalform - welches ja als das Hauptproblem der BOOLEschen Programmierung festgestellt war - stellen sich zwei Fragen: Wodurch wird der einfachste Ausdruck einer BF gebildet und wie wird er fixiert? Die erste Frage führt zu den sogenannten Primimplikanten[1], denn jede einfachste Form eines BOOLEschen Ausdrucks ist die Disjunktion seiner Primimplikanten[2]. Das Problem reduziert sich also auf die Entwicklung von Regeln, welche "produce all prime implicants and only prime implicants."[3]

Innerhalb des Aussagenkalküls werden die Wahrheitswertzuordnungen für die einzelnen Aussagenvariablen ermittelt, welche den ganzen Ausdruck als wahr oder zulässig bestimmen. Diese Zuordnungen bilden die "Lösungen" bei der aussagenlogischen Programmierung. Entsprechend sind alle BOOLEschen Vektoren, die eine bestimmte BF gleich 1 werden lassen, Lösungen dieser Funktion. Der kanonische Ausdruck einer BF stellt somit als Disjunktion aller fundamentalen Produkte, welche diese Funktion erfüllen, eine Lösung dar. Sie ist aber in der Regel nicht die optimale, da sie verbessert werden kann. In Analogie zur linearen Programmierung kann man sie als Nullösung des BOOLEschen Programms bezeichnen. Die optimale Lösung soll wie folgt festgestellt werden[4]: Geht man davon aus, daß jedem Glied einer BF ein nicht-negativer ganzzahliger Wert zugeordnet ist,derart, daß der Wert der Funktion

---

[1] Vgl. Primteiler, Primpolynome in der elementaren Zahlentheorie

[2] Dieses Theorem bildet die allgemeine Grundlage aller Versuche dieser Art und wurde von QUINE formuliert und bewiesen. Vgl. Quine, W.V.: The problem of ..., a.a.O., S. 524. Siehe auch den Beweis bei Balinski, M.L.: Integer Programming ..., a.a.O., S. 284 und Root, James G.: An Application of Symbolic Logic to a Selection problem, in: JORSA, Vol. 12 No. 4 1964, S. 519 ff.

[3] Balinski, M.L.: Integer Programming ..., a.a.O., S. 283.

[4] Vgl. Cobham, A.; Fridshal, R.; North, J.H.: An Application ..., a.a.O., S. 2.

gleich der Summe der Einzelwerte ist, können verschiedene Normalformen gebildet werden, die um so einfacher sind, je geringer ihr Wert ist. Die einfachste Form ist die mit dem kleinsten Wert. Sie ist als die optimale "Lösung" einen BOOLEschen Programms definiert. Zu ihr kommt man durch eine Abfolge immer einfacherer Normalformen. Ist jedem Glied der Funktion bzw. jeder Variablen der gleiche Wert zugeordnet, so ist die optimale Lösung die mit den wenigsten Gliedern bzw. Variablen. Jede minimale Form dieser Art wird aber durch die Primimplikanten gebildet. Ein Produkt P einer BF ist als deren Implikant definiert, wenn zwischen ihnen die implikative Verknüpfung wenn P = 1, dann F = 1 ($P \longrightarrow F = 1$) besteht. Alle kompletten Produkte sind somit stets Implikanten, Implikanten aber nicht notwendigerweise vollständige Produkte. Implikanten vielmehr können wiederum Vielfache anderer Produkte sein, die ihrerseits Implikanten von F sind; z.B. sei für

$$F = ab\bar{c}d + abd$$

das Produkt abd Implikant. Dann ist $ab\bar{c}d$ das $\bar{c}$-fache dieses Implikanten. Implikanten, die selbst nicht mehr ein Vielfaches anderer Implikanten sind - also nicht mehr in Produkte mit weniger Variablen zerlegt werden können, welche die Funktion noch erfüllen - werden als Primimplikanten definiert: "a Prime implicant is a product containing a minimal number of terms which has value 1 only when F has value 1."[1]

(4) Ausgangspunkt des Verfahrens zur Bestimmung der Primimplikanten ist üblicherweise die in eine entwickelte Normalform gebrachte Funktion[2]. Sie besteht aus allen Variablen-Kombinationen, welche entsprechend

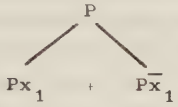
---

[1] Balinski, M.L.: Integer Programming ..., a.a.O., S. 283.
[2] Vgl. Quine, W.V.: The problem ..., a.a.O., S. 524.

dem Funktionsgesetz mit 1 vorherbestimmt sind (d.h. zulässig sind). Gegeben sei z.B. die Funktion

$$F_{(1)} = ab\bar{c}d + abc\bar{d} + a\bar{b}d + \bar{a}bcd + \bar{a}\bar{b}cd + a\bar{c}$$

Die Zerlegungsvorschrift $1 = x_1 + \bar{x}_1$ gibt an, wie jedes unvollständige Glied P, dem eine bestimmte Variable z.B. x_1 fehlt, entsprechend



durch eine endliche Zahl von Operationen[1] entwickelt werden kann. Für F ergibt sich die entwickelte Normalform:

$$F_{(2)} = ab\bar{c}d + ab\bar{c}\bar{d} + a\bar{b}cd + a\bar{b}\bar{c}d + \bar{a}bcd +$$
$$+\bar{a}\bar{b}cd + ab\bar{c}d + a\bar{b}\bar{c}d + ab\bar{c}\bar{d} + a\bar{b}\bar{c}\bar{d}$$

Da doppelte Glieder keinen Einfluß auf den Funktionswert haben, werden sie gestrichen und man erhält als "Nullösung":

$$F_{(3)} = ab\bar{c}d + ab\bar{c}\bar{d} + a\bar{b}cd + a\bar{b}\bar{c}d + \bar{a}bcd$$
$$\bar{a}\bar{b}cd + a\bar{b}\bar{c}\bar{d}$$

Auf diesen Ausdruck ist folgende Vorschrift solange wie möglich anzuwenden[2].

---

[1] Auf diesen leicht einzusehenden Umstand weist FORTET ausdrücklich hin. Vgl. Fortet, R.: L'Algebre de Boole ..., a.a.O., S. 9.

[2] Vgl. Quine, W.V.: The problem..., a.a.O., S. 525, derselbe: A way to simplify Truth Functions, in: AMM Vol. 62 1955, S. 627.
Balinski, M.L.: Integer Programming ..., a.a.O., S. 283.

Haben zwei Glieder ein Element gemeinsam, welches in einem von beiden positiv (= nicht gestrichen), im anderen negativ (= gestrichen) erscheint, so ist die Vereinigung beider Glieder als neues Produkt aufzunehmen. Die Ausgangsglieder sind zu kennzeichnen. Diese Ausgangsglieder sind dabei - soweit möglich - wiederholt zu verschiedenen Kombinationen zu verwenden. Kann diese Regel nicht mehr - auch nicht für die neugeschaffenen Glieder - angewandt werden, sind die nicht gekennzeichneten Glieder als alle möglichen Primimplikanten fixiert. Doppelte Produkte sind wieder zu streichen.

Wie leicht zu sehen ist, kann man auch vom ursprünglichen, nicht entwickelten Ausdruck ($F_{(1)}$) ausgehen. Hierbei ist zu beachten, daß Glieder, die ein Vielfaches anderer darstellen, eliminiert werden, z.B. $ab\bar{c}d$ als das bd-fache von $a\bar{c}$[1]. Aus

		I. Schritt		II. Schritt	
1.	$ab\bar{c}d$*				
		1. $ab\bar{c}$ (1,2)*		$a\bar{c}$ (1,6)	
2.	$ab\bar{c}\bar{d}$*				
		2. $a\bar{c}d$ (1,4)*		$a\bar{c}$ (2,3)	gestrichen, da doppelt
3.	$a\bar{b}cd$*				
		3. $a\bar{c}\bar{d}$ (2,7)*			
4.	$a\bar{b}\bar{c}d$*				
		4. $a\bar{b}d$ (3,4)	———	$a\bar{b}d$	
5.	$\bar{a}bcd$*				
		5. $\bar{b}cd$ (3,6)	———	$\bar{b}cd$	
6.	$\bar{a}b\bar{c}d$*				
		6. $a\bar{b}\bar{c}$ (4,7)*			
7.	$a\bar{b}\bar{c}\bar{d}$*				
		7. $\bar{a}cd$ (5,6)	———	$\bar{a}cd$	

[1] Für das Beispiel ergibt sich dann aus $F_{(T)} = a\bar{b}d + \bar{a}bcd + \bar{a}\bar{b}cd + a\bar{c}$

1. $a\bar{b}d$ — $a\bar{b}d$
2. $\bar{a}bcd$* $\bar{b}cd$ (1,3)
3. $a\bar{b}cd$* $\bar{a}cd$ (2,3)
4. $a\bar{c}$ — $a\bar{c}$

Hierbei muß außerdem beachtet werden: 1. Produkte, aus denen eine Vereinigung gebildet wird, entfallen nur wenn sie vollständig sind. 2. Die Vorschrift $P = Px_1 + P\bar{x}_1$ gilt auch für ungleiche P (vgl. Bildung von $\bar{b}cd$). Daß es sich hier um zwei abweichende Verfahren handelt, ist bislang nicht beachtet worden. Auch QUINE ist dabei nicht einheitlich. Etwas anders noch geht NELSON vor. Vgl. Nelson, Raymond J.: Simplest Normal Truth Functions, in: Journal of Symbolic Logic, Vol. 26 1954, S. 105 ff.

Die Ziffern in den Klammern geben die Produkte an, als deren Vereinigung die neuen Glieder gebildet werden.

Die Menge C = $a\bar{c}$, $a\bar{b}d$, $\bar{b}cd$, $\bar{a}cd$ enthält die Primimplikanten der Funktion F, also

$$F = a\bar{c} + a\bar{b}d + \bar{b}cd + \bar{a}cd$$

Jetzt muß geprüft werden, ob die Anzahl der ermittelten Primimplikanten verringert werden kann. Das ist möglich, wenn zur "Deckung" aller kanonischen Ausdrücke der Funktion nicht alle ermittelten Primimplikanten notwendig sind[1]. Dazu wird entsprechend der sogenannten "prime implicant table" von QUINE[2] eine Matrix A gebildet. Ihre Kopfzeile nimmt die kanonischen Produkte, die Eingangsspalte die Primimplikanten auf[3]. Für die Elemente der Matrix

$$A = \|a_{ij}\|$$

wird gesetzt $a_{ij} = \begin{cases} 1, & \text{wenn der Primimplikant } \alpha_j \text{ den kanonischen Ausdruck } \beta_i \text{ deckt,} \\ 0, & \text{wenn das nicht der Fall ist.} \end{cases}$

---

[1] Unter "Deckung" (im Anglo-Amerikanischen Covering) wird verstanden, daß die Variablenkombination eines Primimplikanten in einem kanonischen Ausdruck in gleicher Weise auftritt, dieser den Primimplikanten also enthält, oder "subsumes" in der Terminologie von QUINE.
Vgl. Quine, W.V.: The Problem ..., a.a.O., S. 526.

[2] Vgl. Quine, W.V.: The Problem of Simplifying ..., a.a.O., S. 526.

[3] Vgl. Cobham, A.; Fridshal, R.; North, J.H.: An Application ..., a.a.O., S. 3. Hier wird diese Matrix als QUINE Matrix einer BOOLEschen Funktion F bezeichnet.

Im Beispiel ergibt sich:

	β \ α	1. $a\bar{c}$	2. $\bar{a}b\bar{d}$	3. $\bar{b}cd$	4. $\bar{a}cd$
1.	$ab\bar{c}d$	1	0	0	0
2.	$ab\bar{c}\bar{d}$	1	0	0	0
3.	$a\bar{b}cd$	0	1	1	0
4.	$a\bar{b}\bar{c}d$	1	1	0	0
5.	$\bar{a}bcd$	0	0	0	1
6.	$\bar{a}\bar{b}cd$	0	0	1	1
7.	$a\bar{b}\bar{c}\bar{d}$	1	0	0	0

Tabelle 9
"QUINE-Matrix" von F

Wie unmittelbar aus der Matrix ersehen werden kann, sind nicht alle Primimplikanten zur "Deckung" des kanonischen Ausdrucks der Funktion notwendig:

β_1, β_2, β_4, β_7 sind durch α_1 abgedeckt,
β_3 ist durch α_2 und α_3 sowie
β_5 und β_6 durch α_4 bzw. α_3 und α_4 gedeckt.

Der Primimplikant α_3 ($\bar{b}cd$) kann aus der Menge C gestrichen werden, da die dazugehörigen kanonischen Produkte schon durch α_2 bzw. α_4 abgedeckt sind. Als einfachste (minimale) Normalform von F ergibt sich:

$$F_{opt} = a\bar{c} + a\bar{b}d + \bar{a}cd$$

Selten sind die QUINE-Matrizen BOOLEscher Programme jedoch

so klein und übersichtlich[1]. Zu ihrer systematischen programmierbaren Vereinfachung sind darum verschiedene Operationen entwickelt worden[2], durch welche alle überflüssigen Primimplikanten eliminiert werden können " ... it was found that these operations resultet in considerable simplifications."[3]

Die "Primimplikanten-Matrix" A einer BOOLEschen Funktion F wird in ihre einfachste Form A* - die reduzierte QUINE Matrix" - überführt, in dem auf sie folgende Operationen in beliebiger Reihenfolge solange wie möglich angewandt werden[4].

(i) Wenn in einer Zeile β_{i_0} nur eine 1 erscheint, z.B. $a_{i_0 j_0} = 1$ und $a_{i_0 j} = 0$ für $j \neq j_0$, so sind die Spalte α_{j_0} und alle Zeilen β_i, deren Elemente $a_{i j_0} = 1$, aus der Matrix zu streichen.

Man setze $x_{j_0} = 1$ (= Kern-Eliminierung)

---

[1] Vgl. Balinski, M.L.; Quandt, R.E.: On an Integer Program ..., a.a.O., S. 300 ff. und denselben Ansatz in: Balinski,M.L.: Integer Programming ..., a.a.O., S. 285.
Balinski beschreibt als Beispiel das Problem der Auslieferung von Waren an verschiedene Kunden (=Deliveryproblem, welches mit BOOLEschen Variablen formuliert werden kann, vgl. S. 43) mit über 388 möglichen Zuordnungen.

[2] Vgl. Quine, W.V.: The Problem ..., a.a.O., S. 527 ff.
derselbe: A way to simplify ..., a.a.O., S. 629 f.
Lobham, A.; Fridshal, R.; North, J.H.: An Application..., a.a.O., S. 4.
Balinski, M.L.: Integer Programming ..., a.a.O., S. 285.

[3] Balinski, M.L.: Integer Programming ..., a.a.O., S. 285.

[4] Regel (i) - "Core-elimination" - und Regel (ii) - "Canonical term elimination" - stammen von Quine, Regel (iii) "prime implicant elimination) von Roth, J.P.: Algebraic Topological Methods for the Synthesis of Switching Systems, I, Transactions of the American Mathematical Society, Vol. 88 1958, S. 301 ff. Zum mathematischen Beweis vgl. Cobham, A.; Fridshal, R.; North, J.H.: An Application ..., a.a.O., S. 18 ff.

(ii) Streichung von kanonischen Ausdrücken:

Wenn eine Zeile β_{i_0} überall da Elemente 1 aufweist, wo auch eine Zeile β_{i_1} die Werte 1 aufweist - d.h. wenn für jedes j, bei dem $a_{i_1 j} = 1$, $a_{i_0 j} = 1$ impliziert wird - , ist die Zeile β_{i_0} zu streichen.

(iii) Streichung von Primimplikanten:

Wenn eine Spalte α_{j_0} nur dort die Werte 1 hat, wo das auch bei einer Spalte α_{j_1} $(j_0 \neq j_1)$ der Fall ist, - d.h. wenn für jedes i, bei dem $a_{i j_0} = 1$, $a_{i j_1} = 1$ impliziert wird - und wenn $b_{j_0} \geq b_{j_1}$ ist, ist Spalte α_{j_0} zu streichen und $x_{j_0} = 0$ zu setzen.

Beispiel: Gegeben sei die Funktion F in der disjunktiven Normalform:

$$F = d\bar{e} + \bar{a}b + a\bar{b}de + ade + \bar{b}\bar{e} + ab\bar{c}e + \bar{b}cde + ac\bar{e}$$

Diese Funktion hat folgende 25 kanonischen Glieder und 12 Primimplikanten:

β_1 : $abcd\bar{e}$	β_{10} : $\bar{a}b\bar{c}de$	β_{19} : $a\bar{b}c\bar{d}\bar{e}$	$\alpha_1 = ad$	$\alpha_7 = b\bar{c}d$
β_2 : $\bar{a}bcd\bar{e}$	β_{11} : $\bar{a}bc\bar{d}e$	β_{20} : $\bar{a}\bar{b}cd\bar{e}$	$\alpha_2 = \bar{a}b$	$\alpha_8 = cd$
β_3 : $a\bar{b}cd\bar{e}$	β_{12} : $\bar{a}b\bar{c}\bar{d}e$	β_{21} : $a\bar{b}\bar{c}\bar{d}\bar{e}$	$\alpha_3 = \bar{a}\bar{e}$	$\alpha_9 = c\bar{e}$
β_4 : $\bar{a}\bar{b}cd\bar{e}$	β_{13} : $\bar{a}bc\bar{d}\bar{e}$	β_{22} : $\bar{a}\bar{b}\bar{c}d\bar{e}$	$\alpha_4 = \bar{e}$	$\alpha_{10} = d\bar{e}$
β_5 : $ab\bar{c}d\bar{e}$	β_{14} : $\bar{a}b\bar{c}\bar{d}\bar{e}$	β_{23} : $ab\bar{c}de$	$\alpha_5 = b$	$\alpha_{11} = b\bar{c}$
β_6 : $\bar{a}b\bar{c}d\bar{e}$	β_{15} : $a\bar{b}cde$	β_{24} : $\bar{a}\bar{b}cde$	$\alpha_6 = bd$	$\alpha_{12} = \bar{b}cd$
β_7 : $a\bar{b}\bar{c}d\bar{e}$	β_{16} : $a\bar{b}\bar{c}de$	β_{25} : $abcd\bar{e}$		
β_8 : $\bar{a}\bar{b}\bar{c}d\bar{e}$	β_{17} : $abcde$			
β_9 : $\bar{a}bcde$	β_{18} : $ab\bar{c}de$			

Tabelle 10

Aus den Primimplikanten bilden wir die BOOLEsche Funktion F, sie ist zu minimieren!

$$F = a\bar{d}x_1 + \bar{a}bx_2 + \bar{a}\bar{e}x_3 + \bar{e}x_4 + bx_5 + bdx_6 + b\bar{c}dx_7$$

$$+ cdx_8 + c\bar{e}x_9 + d\bar{e}x_{10} + b\bar{c}x_{11} + \bar{b}cdx_{12}$$

Offenbar ist die minimale Form für $x_j = 0$ oder 1 von F durch die Glieder fixiert, deren $x_j = 1$ werden.

Primimplikanten-Matrix

	A_0 / α	1	2	3	4	5	6	7	8	9	10	11	12
		ad	$\bar{a}b$	$\bar{a}\bar{e}$	$\bar{e}$	b	bd	$b\bar{c}d$	cd	$c\bar{e}$	$d\bar{e}$	$b\bar{c}$	$\bar{b}cd$
	β												
1	abcd$\bar{e}$	~~1~~	~~0~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~0~~
2	$\bar{a}$bcd$\bar{e}$	0	1	1	1	1	1	0	1	1	1	0	0
3	a$\bar{b}$cd$\bar{e}$	~~1~~	~~0~~	~~0~~	~~1~~	~~0~~	~~0~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~
4	$\bar{a}\bar{b}$cd$\bar{e}$	0	0	1	1	0	0	0	1	1	1	0	1
5	ab$\bar{c}$d$\bar{e}$	~~1~~	~~0~~	~~0~~	~~1~~	~~1~~	~~1~~	~~1~~	~~0~~	~~0~~	~~1~~	~~1~~	~~0~~
6	$\bar{a}$b$\bar{c}$d$\bar{e}$	0	1	1	1	1	1	1	0	0	1	1	0
7	a$\bar{b}\bar{c}$d$\bar{e}$	~~1~~	~~0~~	~~0~~	~~1~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~1~~	~~0~~	~~0~~
8	$\bar{a}\bar{b}\bar{c}$d$\bar{e}$	0	0	1	1	0	0	0	0	0	1	0	0
9	$\bar{a}$bcde	0	1	0	0	1	1	0	1	0	0	0	0
10	$\bar{a}$b$\bar{c}$de	0	1	0	0	1	1	1	0	0	0	1	0
11	$\bar{a}$bc$\bar{d}$e	0	1	0	0	1	0	0	0	0	0	0	0
12	$\bar{a}$b$\bar{c}\bar{d}$e	0	1	0	0	1	0	0	0	0	0	1	0
13	$\bar{a}$bcd$\bar{e}$	0	1	1	1	1	0	0	0	1	0	0	0
14	$\bar{a}$b$\bar{c}\bar{d}\bar{e}$	0	1	1	1	1	0	0	0	0	0	1	0
15	a$\bar{b}$cde	~~1~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~1~~	~~0~~	~~0~~	~~0~~	~~1~~
16	a$\bar{b}\bar{c}$de	~~1~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~
17	abcde	~~1~~	~~0~~	~~0~~	~~0~~	~~1~~	~~1~~	~~0~~	~~1~~	~~0~~	~~0~~	~~0~~	~~0~~
18	ab$\bar{c}$de	~~1~~	~~0~~	~~0~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~0~~	~~0~~	~~1~~	~~0~~
19	abcde	0	0	0	1	0	0	0	0	1	0	0	0
20	$\bar{a}\bar{b}$cd$\bar{e}$	0	0	1	1	0	0	0	0	1	0	0	0
21	a$\bar{b}\bar{c}\bar{d}\bar{e}$	0	0	1	0	0	0	0	0	0	0	0	0
22	$\bar{a}\bar{b}\bar{c}\bar{d}\bar{e}$	0	0	1	1	0	0	0	0	0	0	0	0
23	ab$\bar{c}\bar{d}$e	0	0	0	0	1	0	0	0	0	0	1	0
24	$\bar{a}\bar{b}$cde	0	0	0	0	0	0	0	1	0	0	0	1
25	abcd$\bar{e}$	0	0	0	1	1	0	0	0	1	0	0	0

Tabelle 11

Die Zeilen β_1, β_3, β_5, β_7, β_{15}, β_{16}, β_{17}, β_{18} und Spalte α_1 werden gestrichen wegen (i), $x_1 = 1$ gesetzt.

Primimplikanten-Matrix

A_1

β \ α	2	3	4	5	6	7	8	9	10	11	12
2	1	1	1	1	1	0	1	1	1	0	0
4	0	1	1	0	0	0	1	1	1	0	1
6	1	1	1	1	1	1	0	0	1	1	0
8	0	1	1	0	0	0	0	0	1	0	0
9	1	0	0	1	1	0	1	0	0	0	0
10	1	0	0	1	1	1	0	0	0	1	0
11	1	0	0	1	0	0	0	0	0	0	0
12	1	0	0	1	0	0	0	0	0	1	0
13	1	1	1	1	0	0	0	1	0	0	0
14	1	1	1	1	0	0	0	0	0	1	0
19	0	0	1	0	0	0	0	1	0	0	0
20	0	1	1	0	0	0	0	1	0	0	0
21	0	0	1	0	0	0	0	0	0	0	0
22	0	1	1	0	0	0	0	0	0	0	0
23	0	0	0	1	0	0	0	0	0	1	0
24	0	0	0	0	0	0	1	0	0	0	1
25	0	0	1	1	0	0	0	1	0	0	0

Tabelle 12

Die Zeilen β_2, β_4, β_6, β_8, β_{13}, β_{14}, β_{19}, β_{20}, β_{21}, β_{22}, β_{25} und Spalte α_4 werden gestrichen wegen (i)$x_4 = 1$

A_2	α	2	3	5	6	7	8	9	10	11	12
β											
9		1	0	1	1	0	1	0	0	0	0
10		1	0	1	1	1	0	0	0	1	0
11		1	0	1	0	0	0	0	0	0	0
12		1	0	1	0	0	0	0	0	1	0
23		0	0	1	0	0	0	0	0	1	0
24		0	0	0	0	0	1	0	0	0	1

Tabelle 13

Spalte α_2 streichen wegen (i); $x_2 = 0$

Zeilen β_{11}, β_{12}, β_{23} streichen wegen (ii)

Spalte α_6, α_7, α_{11}, α_{12} streichen wegen (i); $x_6 = x_7 = 0$

Durch diese Streichung ist die Matrix trivial geworden[1] und

[1] Sofern keine Regel mehr angewandt werden kann und die Matrix noch nicht trivial ist (was selten der Fall sein wird), muß mit dem GOMORY Algorithmus weiter gearbeitet werden. Daß man auch nur nach (ii) und (iii) und in jeder beliebigen Reihenfolge zum gleichen Ergebnis kommt, ist leicht nachzuprüfen.

es ergeben sich folgende Zuordnungen:

$$x_3=x_9=x_{10}=x_{11}=x_{12}=0;\ x_5=x_8=1$$

Die vollständige Lösung lautet somit:

$$x_1=x_4=x_5=x_8=1;\ x_2=x_3=x_6=x_7=x_9=x_{10}=x_{11}=x_{12}=0$$

und die minimale Form von F:

$$F_{opt} = ad + \bar{e} + b + cd$$

Wenn mehrere Primimplikanten identische Spalten aufweisen, ist im Hinblick auf das angegebene Minimum-Kriterium derjenige mit den wenigsten Variablen in der Auswahl zu behalten. Das gilt natürlich nur unter der Annahme, daß alle Variablen das gleiche Gewicht[1] haben, d.h. gleichwertig sind. Wird jedem Primimplikanten α_j $(j=1,\ldots,n)$ einer BOOLEschen Funktion F ein bestimmter, nicht-negativer ganzzahliger Wert b_j zugeordnet, dann ist das Problem der Bestimmung der minimalen Form von F - wie es hier dargestellt wurde - absolut gleichbedeutend mit dem Problem, die Werte $x_1, \ldots, x_n$ zu bestimmen, welche den Ausdruck

$$\sum_{j=1}^{n} b_j x_j$$

unter den Beschränkungen

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1$$

[1] "Gewicht" ist hier im übertragenen Sinne als Wertmaßstab zu verstehen.

minimieren, wobei i = 1, ..., m und x_j = 0 oder 1 für j = 1, ..., n[1].

Berücksichtigt man darüber hinaus, daß die Bedingung x_j = 0 oder 1 ersetzt werden kann durch "the equivalent requirement that x_j be a non-negativ integer, the problem is seen to assume the charakter of a straight-forward, if somewhat specialized problem in integer programming."[2] Es ist also durchaus berechtigt, die BOOLEschen Ansätze - wie hier geschehen - als Sonderform der ganzzahligen Programmierung aufzufassen, zumal die entsprechenden Probleme - BALINSKI nennt sie in Anlehnung an graphentheoretische Bezeichnungen "weightet set covering" Probleme-von "considerable interest and importance"[3] sind. Auf sie kommen wir bei der PSEUDO-BOOLEschen Programmierung zurück, welche mit beliebig reellen b_j und a_{ij}-Werten arbeitet.

342. Das BOOLEsche Programm für ein Selektionsproblem

Gegeben seien k Arbeiten A = (a_1, ..., a_k), Arbeiten[4], zu deren Durchführung n Vorrichtungen M = (m_1, ..., m_n) zur Auswahl stehen, deren Kosten durch K = (k_1, ..., k_n) gegeben sind. Es gilt dann

$$KM' \longrightarrow \text{Min.!} \quad (1)$$

---

[1] Vgl. Cobham, A.; Fridshal, R.; North, J.H.: An Application ..., a.a.O., S. 3.

[2] Vgl. Cobham, A.; Fridshal, R.; North, J.H.: An Application ..., a.a.O., S. 3/4. Die Besonderheit wurde von uns mit dem von EICHENBERGER stammenden Ausdruck "Außerordentliche Ganzzahligkeit" gekennzeichnet.

[3] Balinski, M.L.: Integer Programming ..., a.a.O., S. 282. Vgl. auch die Aufzählung auf S. 43 dieser Arbeit.

[4] Vgl. Abschnitt 32 dieses Kapitels und Root, James, G.: An Application of ..., a.a.O., S. 519 ff.

mit $k_i > 0$ und $m_i = 0$ oder 1.

Die Auswahl der Vorrichtungen unterliegt den Beschränkungen $B = (b_1, \ldots, b_l)$, es sind also in Form BOOLEscher Gleichungen die Nebenbedingungen

$$a_i\ (m_1,\ldots,m_n) = 1\ !\ (i=1,\ldots,k) \quad (2)$$

$$\text{und } b_j\ (m_1,\ldots,m_n) = 1\ !\ (j=1,\ldots,l) \quad (3)$$

zu formulieren. Da die Lösungsvektoren des Problems nur die Entscheidungen enthalten, für die sowohl (2) als auch (3) gleich 1 werden, kann der kanonische Ausdruck

$$A^0 = \overset{k}{\underset{i=1}{//}}\ a_i \overset{l}{\underset{j=1}{//}}\ b_j$$

gebildet werden und man erhält die aggregierte Bedingung

$$A^*\ (m_1,\ldots,m_n) = 1\ ! \quad (4)$$

Es ist die Durchführung von 3 Aufgaben a_1, a_2 und a_3 geplant, insgesamt stehen 6 verschiedene Vorrichtungen zur Wahl: $m_1 = 1{,}2$, $m_2 = 2$, $m_3 = 0{,}6$, $m_4 = 3{,}1$, $m_5 = 1{,}7$ und $m_6 = 2{,}3$ GE. Für die einzelnen Arbeiten gelten die Durchführungsmöglichkeiten[1]:

$$a_1 = m_3m_5m_6 + m_3m_4 + m_2m_5 + m_4m_5$$
$$a_2 = m_4m_5m_6 + m_1m_3m_4 + m_3m_6 + m_3m_5$$
$$a_3 = m_4 + m_1m_5 + m_2m_6$$

[1] Arbeit 1 kann durch die Maschinenkombinationen m_3, m_5 und m_6 oder m_3 und m_4 oder m_2 und m_5 oder m_4 und m_5 erledigt werden.

Der Einsatz der Maschinen unterliegt den räumlichen bzw. technischen und finanziellen Beschränkungen:

$$b_1 = \bar{m}_2\bar{m}_4\bar{m}_6$$

$$b_2 = \bar{m}_1\bar{m}_3\bar{m}_6$$ [1]

Als kanonischer Ausdruck A* ergibt sich, wie aus Tabelle 14 zu ersehen:

$$A^* = \bar{m}_1\bar{m}_2\bar{m}_3m_4m_5m_6 + \bar{m}_1\bar{m}_2m_3m_4\bar{m}_5m_6 + \ldots + m_1m_2m_3m_4m_5\bar{m}_6$$

Da der Rechengang im vorigen Abschnitt ausführlich dargestellt worden ist, wird hier auf die Wiedergabe der Bildung der Primimplikanten und die Vereinfachung der dazugehörigen Matrix verzichtet. Es ergeben sich folgende 3 logisch möglichen, im BOOLEschen Programm optimale Maschinenkombinationen:

$$A^0 = m_4m_5m_6 + m_3m_4m_5 + m_2m_3m_5m_6$$

Diese Menge der Primimplikanten "is the smallest complete set that can defined without some knowledge of the constants, k_i, in the criterion function, (1)" [2]

Der Verfahrensvergleich unter Berücksichtigung numerischer b_j-Werte bestimmt die Maschinenkombination $m_3m_4m_5$ als optimal.

Sicher kann diese Lösung ohne Schwierigkeiten allein durch gedankliches Durchprobieren der logischen Möglichkeiten gefunden werden. Diesem Einwand ist die bereits bei der aussagenlogischen Programmierung angedeutete Feststellung ent-

---

[1] b_2 besagt z.B.: Die Maschinen 1, 3 und 6 können nicht gleichzeitig berücksichtigt werden.

[2] Root, J.G.: An Application ..., a.a.O., S. 523.

Nr.	m_1	m_2	m_3	m_4	m_5	m_6	a_1	a_2	a_3	b_1	b_2	A*	
1	0	0	0	0	0	0	0	0	0	1	1	0	
2	0	0	0	0	0	1	0	0	0	1	1	0	
3	0	0	0	0	1	0	0	0	0	1	1	0	
4	0	0	0	0	1	1	0	0	0	1	1	0	
5	0	0	0	1	0	0	0	0	1	1	1	0	
6	0	0	0	1	0	1	0	0	1	1	1	0	
7	0	0	0	1	1	0	1	0	1	1	1	0	
8	0	0	0	1	1	1	1	1	1	1	1	1	$= \bar{m}_1\bar{m}_2\bar{m}_3m_4m_5m_6$
9	0	0	1	0	0	0	0	0	0	1	1	0	
10	0	0	1	0	0	1	0	1	0	1	1	0	
11	0	0	1	0	1	0	0	1	0	1	1	0	
12	0	0	1	0	1	1	0	1	0	1	1	0	
13	0	0	1	1	0	0	1	0	1	1	1	0	
14	0	0	1	1	0	1	1	1	1	1	1	1	$= \bar{m}_1\bar{m}_2m_3m_4\bar{m}_5m_6$
15	0	0	1	1	1	0	1	1	1	1	1	1	$= \bar{m}_1\bar{m}_2m_3m_4m_5\bar{m}_6$
16	0	0	1	1	1	1	1	1	1	1	1	1	$= \bar{m}_1\bar{m}_2m_3m_4m_5m_6$
17	0	1	0	0	0	0	0	0	0	1	1	0	
18	0	1	0	0	0	1	0	0	1	1	1	0	
19	0	1	0	0	1	0	1	0	0	1	1	0	
20	0	1	0	0	1	1	1	0	1	1	1	0	
21	0	1	0	1	0	0	0	0	1	1	1	0	
22	0	1	0	1	0	1	0	0	1	0	1	0	
23	0	1	0	1	1	0	1	0	1	1	1	0	
24	0	1	0	1	1	1	1	1	1	0	1	0	
25	0	1	1	0	0	0	0	0	0	1	1	0	
26	0	1	1	0	0	1	0	1	1	1	1	0	
27	0	1	1	0	1	0	0	1	0	1	1	0	
28	0	1	1	0	1	1	1	1	1	1	1	1	$= \bar{m}_1m_2m_3\bar{m}_4m_5m_6$
29	0	1	1	1	0	0	1	0	1	1	1	0	
30	0	1	1	1	0	1	1	1	1	0	1	0	
31	0	1	1	1	1	0	1	1	1	1	1	1	$= \bar{m}_1m_2m_3m_4m_5\bar{m}_6$
32	0	1	1	1	1	1	1	1	1	0	1	0	
33	1	0	0	0	0	0	0	0	0	1	1	0	
34	1	0	0	0	0	1	0	0	0	1	1	0	

(I)

35	1	0	0	0	1	0	0	0	1	1	1	0	
36	1	0	0	0	1	1	0	0	1	1	1	0	
37	1	0	0	1	0	0	0	0	1	1	1	0	
38	1	0	0	1	0	1	0	0	1	1	1	0	
39	1	0	0	1	1	0	1	0	1	1	1	0	
40	1	0	0	1	1	1	1	1	1	1	1	1	$= m_1\bar{m}_2\bar{m}_3m_4m_5m_6$
41	1	0	1	0	0	0	0	0	0	1	1	0	
42	1	0	1	0	0	1	0	1	0	1	0	0	
43	1	0	1	0	1	0	0	1	1	1	1	0	
44	1	0	1	0	1	1	1	1	1	1	0	0	
45	1	0	1	1	0	0	1	1	1	1	1	1	$= m_1\bar{m}_2m_3m_4\bar{m}_5\bar{m}_6$
46	1	0	1	1	0	1	1	1	1	1	0	0	
47	1	0	1	1	1	0	1	1	1	1	1	1	$= m_1\bar{m}_2m_3m_4m_5\bar{m}_6$
48	1	0	1	1	1	1	1	1	1	1	0	0	
49	1	1	0	0	0	0	0	0	0	1	1	0	
50	1	1	0	0	0	1	0	0	1	1	1	0	
51	1	1	0	0	1	0	1	0	1	1	1	0	
52	1	1	0	0	1	1	1	0	1	1	1	0	
53	1	1	0	1	0	0	0	0	1	1	1	0	
54	1	1	0	1	0	1	0	0	1	0	1	0	
55	1	1	0	1	1	0	1	0	1	1	1	0	
56	1	1	0	1	1	1	1	1	1	0	1	0	
57	1	1	1	0	0	0	0	0	0	1	1	0	
58	1	1	1	0	0	1	0	1	1	1	0	0	
59	1	1	1	0	1	0	1	1	1	1	1	1	$= m_1m_2m_3\bar{m}_4m_5\bar{m}_6$
60	1	1	1	0	1	1	1	1	1	1	0	0	
61	1	1	1	1	0	0	1	1	1	1	1	1	$= m_1m_2m_3m_4\bar{m}_5\bar{m}_6$
62	1	1	1	1	0	1	1	1	1	0	0	0	
63	1	1	1	1	1	0	1	1	1	1	1	1	$= m_1m_2m_3m_4m_5\bar{m}_6$
64	1	1	1	1	1	1	1	1	1	0	0	0	

(II)

Tabelle 14

gegenzuhalten, daß bei umfangreichen Problemen allein systematische Rechenoperationen zum Ziel führen, die darüberhinaus leicht programmierbar sind. Wie gezeigt wurde, können spezielle ganzzahlige Programmierungsaufgaben nach entsprechender Aufspaltung der Variablen als BOOLEsche Programme formuliert und mit den angegebenen Operationen gelöst werden. Die Lösungen sind durch die reduzierten QUINE-Matrizen fixiert. Tatsächlich haben praktische Erfahrungen bei der automatischen Durchrechnung ganzzahliger Programmierungsaufgaben gezeigt "... that integer programming is an effective means of dealing with the minimization of BOOLEan function problems."[1]

35. BOOLEsche Programme in Verbindung mit der Theorie der Graphen

351. Die Graphendarstellung in betriebswirtschaftlichen Modellen

Das Kapitel über die sogenannte BOOLEsche Programmierung soll nicht abgeschlossen werden, ohne auf einen weiteren interessanten Aspekt dieser Modelle hinzuweisen. Ein geeignetes Hilfsmittel zur anschaulichen Darstellung vieler betriebswirt-

---

[1] Balinski, M.L.: Integer Programming ..., a.a.O., S. 305. Technische Einzelheiten der Durchrechnung BOOLEscher Programme auf Elektronenrechnern sind hier nicht zu erörtern. Eine ausführliche Programmbeschreibung findet sich z.B. bei Cobham, A.; Fridshal, R.; North, J.H.: An Application ..., a.a.O., S. 22 ff. Zu weiteren Einzelheiten s.a. Cobham, A.; North, J.M.: Extensions of Integer Programming ..., a.a.O., S. 6 ff.

schaftlicher Probleme ist die Abbildung durch Graphen[1]. Diese wiederum können durch BOOLEsche Ausdrücke beschrieben werden, welche die Anwendung BOOLEscher Rechenoperationen möglich machen. Unter einem Graphen oder Streckenkomplex $G = [N, A]$ versteht man eine Menge von n Punkten $N = \{x_1, \ldots, x_n\}$ und eine Menge $A = \{x_i, x_j\}$ von Paaren dieser Punkte, d.h. $x_i, x_j \in N^2$.

Die geordneten Paare der Punkte oder Knoten oder Ecken grenzen innerhalb des Graphen Teilstrecken ab, welche als Kanten, Bogen oder Zweige des Graphen bezeichnet werden. Diese Kanten sind entweder gerichtet oder ungerichtet, ihre Anzahl ist endlich oder unendlich. Jedem Graphen kann auf eindeutige Weise eine Matrix zugeordnet werden[3], die sogenannte Knoten-

---

[1] Vgl. z.B. Angermann, A.: Entscheidungsmodelle, a.a.O., S.39. Mayhugh, J.O.: On the mathematical theory of schedules, in: MS, Vol. 11 No. 2 1964, S. 289 ff. und besonders Debry, Robert et Huyberechts, Simone: Quelques Aspects de la Theorie des Graphes, in: Cahiers Du Centre D'Etudes De Recherche Operationnelle, No. 2 1959, S. 29 ff.
Harary, Frank: Graph Theoretic Methods in the Management Sciences, in: MS Vol. 5 No. 4 1959, S. 387 ff.
Avondo-Bodino, G.: Economic Applications of the Theory of Graphs, New York 1962.
Notwendig ist an dieser Stelle auch ein Hinweis auf die Hilfsmittel zur Termin- und Ablaufplanung wirtschaftlicher Aktivitäten, welche unter den verschiedensten Bezeichnungen - wie PERT - CPM - RAMPS usw. - als sogenannte Netzwerktechniken Eingang in die Betriebswirtschaftslehre gefunden haben und ebenfalls auf Graphen zurückgehen. Vgl. dazu die erschöpfende Aufzählung bei Tempel, K.-H.: Wartezeitmodelle, a.a.O., S. 236 ff. sowie Algan, Michel: Reihenfolgeprobleme und Graphentheorie, in: Ufo Bd. 8 Heft 2 1964, S. 53 ff.

[2] Zur Theorie der Graphen vgl. König, D.: Theorie der endlichen und unendlichen Graphen, New York 1950.
Berge, Claude: The Theory of Graphs and its Applications, London - New York 1962.
Ore, Oystein: Graphs and their Uses, New York 1963.

[3] Vgl. hierzu Wenke, K.: Matrizenmodelle in der Großindustrie, in: Anwendungen der Matrizenrechnung auf wirtschaftliche und statistische Probleme, Hrsg. Adam, A.; Ferschl, F. u.a. Würzburg 1963, S. 112 ff., bes. S. 117 ff.

Inzidenz-oder einfach Inzidenz-Matrix. Die Zeilen der Matrix nehmen die Knoten, ihre Spalten die Kanten auf. Matrixelemente sind BOOLEsche Zahlen, und zwar steht an der Stelle a_{ij} eine 1, wenn Knoten i und Kante j zusammengehören - inzidieren - , in allen anderen Fällen eine 0.

Eine derartige Matrix, welche mit MAGHOUT[1] auch als BOOLEsche Matrix bezeichnet werden kann, bildet jede Struktur, zwischen deren Elementen binäre Beziehungen bestehen, in gleicher Weise eineindeutig ab, wie der dazugehörige Graph. BOOLEsche Matrix und Graph sind äquivalente Abbildungen[2]. Es ergibt sich, daß Probleme, welche durch Graphen abgebildet werden können, gleichermaßen als BOOLEsche Programme formuliert werden können. Hierbei sind allerdings zwei typische Fälle zu unterscheiden, wenn man - wie erforderlich - auf die zugrundeliegenden Probleme zurückgeht:

(1) Die Graphendarstellung beschränkt sich auf Aussagen über Anzahl bzw. Art der Knoten und Kanten, d.h. auf die Wiedergabe einer statischen Struktur.

(2) Zusätzlich zu (1) sind Angaben über Prozesse gemacht, welche innerhalb des Streckenkomplexes ablaufen; z.B. über Ströme mit Stärke und Richtung, Kapazität der Kanten, Ursprungs- und Mündungspunkte von Strömen. Im Vordergrund steht in diesem Fall der zwischen den Aktivitäten bestehende Wirkzusammenhang.

---

[1] Vgl. Maghout, Khaled: Applications de l'Algebre de Boole a la Theorie des Graphes et aux programmes lineaires et quadratiques, in: Cahiers Du Centre D'Etudes De Recherche Operationnelle, Vol. 5 1963, S. 21 ff.

[2] "It is well established that homogeneous binary relations defined on finite sets of elements can be represented in 1-1 fashion by means of two equivalent formalismus: (i) by finite Boolean relation matrices of zeros and ones; and (ii) by finite directed graphs."
Rosenblatt, David: On Some Aspects of Models of Complex Behavioral Systems, in: Machol, R.E.: Information and Decision Processes, New York - Toronto - London 1960, S. 63.

Ein Graph, in dem entsprechend (2) ein Strom, oder allgemein "Masse" von einem Ursprungspunkt (Quelle, source) zu einem Endpunkt (Mündung, sink) fließt, wird als Netzwerk bezeichnet[1]. Netzwerke sind hier als besondere Formen mathematischer Programme definiert, welche bei Fragen der Güterverteilung, des Transportes und besonders bei der Ermittlung der maximalen Leistungsfähigkeit von Verbundsysthemen aller Art auftreten. Sind hierbei die Kapazitäten der Kanten (= Leitungen, Kanäle) beschränkt, wird man auf die Probleme der sogenannten FORD - FULKERSON - Theorie[2] geführt.

Die klassische Fragestellung[3] dieser Probleme ist die nach dem maximalen (bzw. minimalen) Fluß durch ein Netzwerk[4].

---

[1] Vgl. die Definition bei Hadley, G.: Linear Programming, Reading - Palo/Alto - London 1963, S. 334.

[2] Vgl. Ford, L.R.; Fulkerson, D.R.: Maximal Flow through a Network, in: Canadian Journal of Mathematics, Vol. VIII, 1956, S. 399 ff.
Ford, L.R.; Fulkerson, D.R.: A Simple Algorithm for Finding Maximal Network Flows and an Application to the Hitchcock Problem, in: Canadian Journal of Mathematics, Vol. IX 1957, S. 210 ff.
Ford, L.R.; Fulkerson, D.R.: A Suggested Computation for Maximal Multi-Commodity Network Flows, in: MS, Vol. 5 No. 1 1958, S. 97 ff.
Fulkerson, D.R.: Increasing the Capacity of a Network: The Parametric Budget Problem, in: MS, Vol. 5 No. 4 1959, S. 472 ff.
Fulkerson, D.R. and Dantzig, G.B.: Computations of Maximal Flows in Networks, in: Naval Research Logistics Quarterly, Vol. 2 No. 4 1955, S. 277 ff.
Fulkerson, D.R.: Flows in Networks, in: Recent Advances in Mathematical Programming, Hrsg.: Graves, Robert L. and Wolfe, Philip, New York - Toronto - London 1963, S. 319 ff.
Ford, L.R.; Fulkerson, D.R.: Flows in Networks, Princeton 1965.

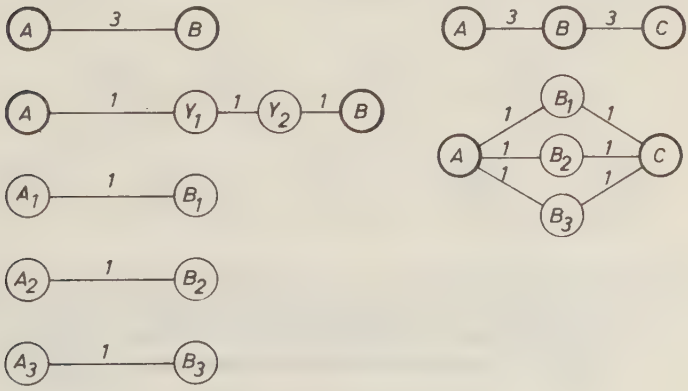
[3] Vgl. zur geschichtlichen Entwicklung dieses Fragenkomplexes Fulkerson, D.R.: Flows in Networks, a.a.O., S. 319 und Dantzig, G.B.: Lineare Programmierung und Erweiterungen, a.a.O., S. 343 ff.

[4] Als weiteres Netzwerk-Problem hat ausgehend vom allgemeinen Transportmodell die Frage nach dem kürzesten Weg und das sogenannte Umlade-Problem zahlreiche Interpretationen erfahren. Vgl. Dantzig, G.B.: Lineare Programmierung und Erweiterungen, a.a.O., S. 400 ff. und Dantzig, G.B.: On the Shortest Route through a Network, in: MS, Vol. 6 No. 2 1960, S. 187 ff.

Daß hierfür durch PSEUDO-BOOLEsche Ansätze ein neues und zweckmäßiges Lösungsverfahren entwickelt werden kann, wird im folgenden Kapitel gezeigt. BOOLEsche Programme sind zur Abbildung und Lösung von Netzwerkproblemen nicht geeignet, sie sind auf Fragestellungen beschränkt, welche mit den unter (1) angegebenen Daten formuliert werden können. Diese Einschränkung beruht nicht so sehr darauf, daß Netzwerke "gewichtete" Kanten haben - diese könnten ja in einwertige Verbindungslinien aufgespalten und in einer erweiterten Matrix erfaßt werden[1], als in der mathematischen Eigenart der Minimierung BOOLEscher Funktionen.

---

[1] Vgl. die Aufspaltung von x_i in $y_{i1} + y_{i2} + \ldots$ mit $y_{ij} = 0$ oder 1. im 3. Kapitel, denn " ... one could introduce sufficiently many additional vertices on the branches of the graph so that the length of each section identified as a separate branch is equal to one",
Hakimi, S.L.: Optimum Distribution of Switching Centers in a Communication Network and some related Graph theoretic problems, in: JORSA, Vol. 13 No. 3 1965, S. 463. "Gewichtet" werden die Kanten durch besondere Angaben über Länge und Kapazität. Eine Aufspaltung der Aktivitäten kann durch die Einführung zusätzlicher Ereignisse erreicht werden, z.B. statt

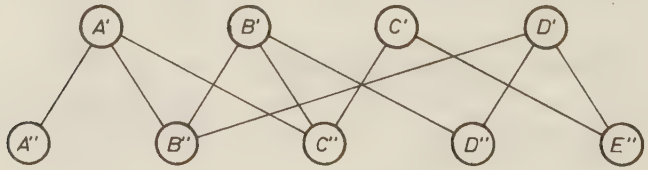


352. BOOLEsche Programme für Probleme, welche durch Graphen abgebildet werden können

Sofern betriebswirtschaftliche Entscheidungsprobleme durch die Beschreibung ihrer Struktur allein gemäß (1) abgebildet werden können, ist ihre Lösung hinsichtlich einiger interessanter Fragestellungen möglich. Diese können unter folgende graphentheoretische Ansätze subsumiert werden[1]:

---

[1] Bekannt ist in diesem Zusammenhang das klassische Problem der Graphentheorie, die Bestimmung der sog. "Chromatic number" eines Graphen, eine Fragestellung, welche auch für ökonomische Überlegungen relevant ist, etwa bei der Abgrenzung wirtschaftlicher Interessengebiete. Sie entsteht bei der Mehrfarben-(=Chromatic)kennzeichnung von Landkarten mit einer minimalen Anzahl verschiedener Farben unter der Bedingung, daß gleichfarbige Flächen nicht benachbart sind.
Vgl. hierzu Berge, Cl.: The Theory of Graphs ..., a.a.O., S. 30 ff. und Ore, O.: Graphs and their uses, a.a.O., S. 109 ff.
Denkt man schließlich an den Sonderfall eines "bipartite" Graphen, so hat man das äquivalente Abbild der Problemstruktur, welche den wichtigen Assignationsproblemen zugrunde liegt. "Bipartite", zweiteilig heißt ein Graph, dessen Menge N der Knoten in zwei wohlunterschiedene Teilmengen N' und N" (z.B. Arbeiter und Arbeiten darstellend) zerfällt, wobei jede Kante aus G einen Punkt $n' \in N'$ und einen Punkt $n'' \in N''$ verbindet, z.B.



Die Kanten repräsentieren mögliche Zuordnungen.
Vgl. hierzu Ivanescu, P.L.; Rudeanu, S.: A Pseudo-Boolean Approach to Matching Problems in Graphs with Applications to Assignment and Transportation Problems, unveröffentlichtes Manuskript, Bukarest 1966.

1. Welches ist die minimale Teilmenge n aus der Menge N der Knoten des Graphen G, die alle Kanten abdeckt, d.h. zu deren Elementen alle Kanten des Graphen inzident sind?

   Welches ist die minimale Teilmenge a aus der Menge A der Kanten des Graphen, die alle Knoten abdeckt?

2. Welches ist die maximale Teilmenge n' aus der Menge N der Knoten von G unter der Bedingung, daß keine zwei Elemente von n' zu ein und derselben Kante inzident sind?

   Welches ist die maximale Teilmenge a' aus der Menge A der Kanten des Graphen G unter der Bedingung, daß keine zwei Kanten inzident sind?

Bekannt sind diese Ansätze unter ihren anglo-amerikanischen Bezeichnungen als Minimum -Cover bzw. Maximum-Matching-Probleme[1]. Sie können auch als ganzzahlige Programme interpretiert werden, z.B. gilt für 1.:

$$\sum_{j=1}^{n} x_j \longrightarrow \text{Min.!}$$

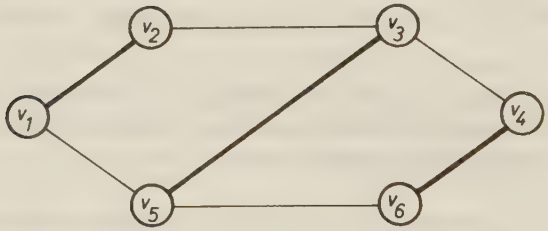
$$\text{mit} \quad \sum_{j=1}^{n} a_j \, x_j \geq \alpha_0 \qquad x_j = 0 \text{ oder } 1,$$

wobei a_0 ein Spaltenvektor mit m Einsen ist und $a_j = (a_{1j}, \ldots, a_{mj})^T$ die Spalten der Inzidenz-Matrix A

---

[1] Vgl. Ore, O.: Graphs and their Uses, a.a.O., S. 43 ff.
Berge, Cl.: The Theory of Graphs ..., a.a.O., S. 77 und 92 ff.
Balinski, M.L.: Integer Programming ..., a.a.O., S. 276 ff.
Hakimi, S.L.: Optimum Distribution of Switching Centers ..., a.a.O., S. 467 ff.

des Graphen G sind[1], x_j steht hier für die Knoten bzw. Kanten, welche durch die Spalten der Matrix dargestellt sind und deren Anzahl unter Beachtung der Nebenbedingung minimiert werden soll. Es ist leicht einzusehen, daß die Anzahl der Kanten in einem "minimalen cover" und "maximalen matching" gleich ist[2]. In Fig. 5 z.B. sind mit $g_1 = \{(v_1, v_2), (v_3, v_5), (v_4, v_6)\}$ 3 Kanten sowohl für 1. als auch für 2. fixiert[3].



Figur 5

Mit Hilfe graphentheoretischer Sätze können die skizzierten Modelle zwar schon seit längerer Zeit gelöst werden[4], doch diese Verfahren sind oft sehr langwierig, schneller kommt man über die Minimierung BOOLEscher Funktionen zum Ziel. Diese Kombination graphentheoretischer Sätze mit BOOLEschen Operationen ist neu und " ... the usefulnes of constructing

---

[1] Vgl. Balinski, M.L.: Integer Programming ..., a.a.O., S. 276.

[2] Für 2. gilt $\sum_{j=1}^{n} x_j \longrightarrow$ Max.! mit $\sum_{j=1}^{n} \alpha_j x_j \leq \alpha_0$, $x_j = 0$ oder 1.

[3] Eine andere Möglichkeit wäre z.B. $g_2 = \{(v_1, v_5), (v_2, v_3), (v_4, v_6)\}$

[4] Vgl. Balinski, M.L.: Integer Programming ..., a.a.O., S. 277 ff., der einige Verfahren beschreibt. Da hier nur BOOLEsche Verfahren interessieren, kann auf eine Wiedergabe der graphentheoretischen Methoden verzichtet werden.

Boolean functions over a graph G ..." ergibt sich z.B. daraus " ... that functions may be devised that generate all matchings ..."[1].

353. BOOLEsche Interpretation eines graphentheoretischen Modells

Ausgangspunkt der Überlegungen sei ein Beispiel: Ein Unternehmen hat innerhalb seines gesamten Absatzgebietes 12 Vertretungen. Von diesen sollen gerade soviel zu Stützpunkten (Niederlassungen) ausgebaut und mit umfangreichen Lägern ausgerüstet werden, daß die Entfernung d von den Vertretungen zu den Niederlassungen höchstens gleich eins ist. Interpretiert man die Vertretungen als die Knoten des Streckenkomplexes mit N_i und mit n_k deren Teilmenge, welche als Stützpunkte der Entfernungsbedingung genügen, ergibt sich die Beziehung

$$d\,(N_i, n_k) \leqq 1 \qquad (i=1,\ldots,n)$$

Durch die Entwicklung einer BOOLEschen Funktion über den Graphen werden alle Teilmengen von Knoten n_k fixiert, welche die Bedingung erfüllen. Sei n^i eine beliebige dieser Teilmengen, d.h.

$$d\,(N_i, n^i) \leqq 1$$

n^i ist dann diejenige Teilmenge aller Punkte des Graphen, welche zu einem bestimmten Knoten N_i inzident sind, d.h.

---

[1] Hakimi, S.L.: Optimum Distribution of Switching Centers ..., a.a.O., S. 463 f.
Vgl. hierzu vor allem auch Rudeanu, S.; Ivanescu, P.L.: A Pseudo-Boolean Approach ..., a.a.O., s. S. 1: "Many a problem of graph theory can be formulated and solved as pseudo-boolean programs."

die von dem Punkt N_i höchstens durch eine Kantenlänge getrennt sind. Wird jeder Punkt N_i mit einer BOOLEschen Variablen x_i bewertet, und die BOOLEsche Summe dieser Variablen, welche den Elementen in n^i zugeordnet sind, mit X_i bezeichnet, ergibt sich die gesuchte BF als BOOLEsches Produkt der BOOLEschen Summen X_i aller Variablen x_i:

$$f(x_1, \ldots, x_n) = \prod_{i}^{1,n} X_i \quad [1]$$

Da diese Funktion nur und nur dann gleich 1 wird, wenn jeder Faktor X_i gleich 1 ist - wobei die Summe X_i gemäß

$$x_1N_1 + x_2N_2 + \ldots x_nN_n = \bigvee_{i}^{1,n} x_iN_i$$

zerlegt und für

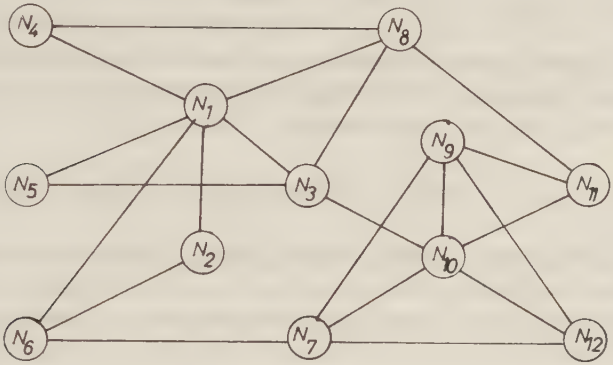
$$N_i = \begin{cases} 0, & \text{wenn } N_i \notin n^i \\ 1, & \text{wenn } N_i \in n^i \end{cases}$$

gesetzt wird - ist für den Fall $f(x_1, \ldots, x_n) = 1$ fixiert, daß alle logisch möglichen Abdeckungen berücksichtigt sind.

---

[1] Vgl. auch Hakimi, S.L.: Optimum Distribution of Switching Centers ..., a.a.O., S. 467. H. spricht von einer "Covering-function".
Es ist zu beachten, daß dieser Boolesche Ausdruck der oben definierten konjunktiven Normalform entspricht, während der in 341. dargestellte der disjunktiven Normalform entspricht.

Unserem Beispiel liege die in Figur 6 wiedergegebene Struktur zugrunde.



Figur 6

Als Überdeckungsfunktion ergibt sich dann:

$$f(x_1,\ldots,x_n)=(x_1+x_2+x_3+x_4+x_5+x_6+x_8)(x_2+x_1+x_6)(x_3+x_1+x_5+x_8+x_{10})$$

$$(x_4+x_1+x_8)(x_5+x_1+x_3)(x_6+x_1+x_2+x_7)(x_7+x_6+x_9+x_{10}+x_{12})$$

$$(x_8+x_1+x_3+x_4+x_{11})(x_9+x_7+x_{10}+x_{11}+x_{12})(x_{10}+x_3+x_7+x_9+x_{11}+x_{12})$$

$$(x_{11}+x_8+x_9+x_{10})(x_{12}+x_7+x_9+x_{10})$$

Zur Vereinfachung werden die Teilmengen n^i, die alle Elemente einer kleineren Teilmenge $n^{i'}$ enthalten, d.h. $n^{i'} \subset n^i$, gestrichen. Der Wert der Funktion bleibt unverändert[1].

[1] Vgl. hierzu die Ausführungen über Primimplikanten, die analog gelten und den Beweis des Rechengangs bei Hakimi, S.L.: Optimum Distribution ..., a.a.O., S. 467 f.

$$f(x_1,x_2,\ldots,x_n)=(x_2+x_1+x_6)(x_4+x_1+x_8)(x_5+x_1+x_3)(x_{11}+x_8+x_9+x_{10})$$

$$(x_{12}+x_7+x_9+x_{10})$$

Unter Beachtung der mathematischen Gesetze über die Distribution und Absorption von BOOLEschen Ausdrücken[1] ergibt sich - geordnet nach Zahl der Elemente:

$$f(x_1,x_2,\ldots,x_n)=x_1x_9+x_1x_{10}+x_1x_7x_8+x_1x_7x_{11}+x_1x_8x_{12}+x_1x_{11}x_{12}$$

$$+x_2x_3x_4x_9+x_2x_3x_4x_{10}+x_2x_4x_5x_9+x_2x_4x_5x_{10}+x_3x_4x_6x_9+x_3x_4x_6x_{10}$$

$$+x_3x_6x_7x_8+x_3x_6x_8x_9+x_3x_6x_8x_{10}+x_3x_6x_8x_{12}+x_4x_5x_6x_9+x_4x_5x_6x_{10}$$

$$+x_4x_5x_7x_8+x_5x_6x_8x_9+x_5x_6x_8x_{10}+x_5x_6x_8x_{12}+x_2x_3x_4x_7x_8+x_2x_3x_4x_7x_{11}$$

$$+x_2x_3x_4x_8x_{10}+x_2x_3x_4x_{11}x_{12}+x_2x_3x_4x_8x_{12}+x_2x_4x_5x_7x_8+x_2x_4x_5x_7x_{11}$$

$$+x_2x_4x_5x_8x_{12}+x_2x_4x_5x_{11}x_{12}+x_3x_4x_6x_7x_{11}+x_3x_4x_6x_{11}x_{12}+x_4x_5x_6x_7x_{11}$$

$$+x_4x_5x_6x_{11}x_{12}$$

Somit sind 35 mögliche "Abdeckungen" ermittelt, die der gestellten Bedingung genügen und von denen $n_{1,2}=\{(N_1,N_9),(N_1,N_{10})\}$ optimal sind[2].

---

[1] Vgl. Abschnitt 212.

[2] Sicher kann dieses Ergebnis bei relativ kleinem i unmittelbar aus dem Graphen abgelesen werden, für komplexe Graphen ist ein systematisches Verfahren jedoch unerläßlich. Seine Bedeutung mag veranschaulichen, daß mit entsprechenden Programmen erste und gute Erfahrungen bei der praktischen Durchrechnung auf Elektronenrechnern gesammelt worden sind.
Vgl. Balinski, M.L.: Integer Programming ..., a.a.O., S.304 f. und Hakimi, S.L.: Optimum Distribution ..., a.a.O., S. 469: "... there are efficient digital computer programs for such a purpose ..."

Der spezielle Ansatz mit der Bedingung $d(N_i, n_k) \leqq 1$ kann ohne Schwierigkeiten zu dem allgemeinen Fall erweitert werden, in dem eine Teilmenge n^i von Knoten ermittelt wird, deren Elemente N_j zu einem Knoten N_i die Entfernung d haben, wobei d jeden beliebigen ganzzahligen Wert annehmen kann (beliebig unter Beachtung der Mächtigkeit des Graphens). Die entsprechende Bedingung lautet[1]:

$$n^i = \left\{ N_j \,\middle|\, d(N_j, N_i) \leqq d \right\}$$

Da die Berechnung einer "Abdeckung" durch eine maximale Anzahl von Kanten (matching-problem) analog durchzuführen ist, wird auf eine Darstellung verzichtet[2]. Desgleichen soll auf einige nützliche, aber mehr am Rande liegende Operationen mit BOOLEschen Matrizen im Zusammenhang mit wirtschaftswissenschaftlichen Modellen an dieser Stelle nur verwiesen werden[3].

---

[1] Vgl. Hakimi, S.L.: Optimum Distribution ..., a.a.O., S. 469 und Balinski, M.L.: Integer Programming ..., a.a.O., S. 281 f. und
Rudeanu, S.; Ivanescu, P.L.: A Pseudo-Boolean Approach..., a.a.O.

[2] Ein anschauliches Beispiel bringt Hakimi, S.L.: Optimum Distribution ..., a.a.O., S. 471 f.

[3] Vgl. Kemeny, J.G.; Snell, J.L.; Thompson, G.L.: Einführung in die endliche Mathematik, a.a.O., S. 311 ff.
Wenke, K.: Matrizenmodelle in der Großindustrie, in: Anwendungen der Matrizenrechnung auf wirtschaftliche und statistische Probleme, hrsgg. von Adam, A.; Ferschl, R. u.a., 2. Aufl. Würzburg 1963, S. 117 ff.
Fortet, R.: Applications de l'Algebre ..., a.a.O., S. 18 f.

VIERTES KAPITEL

DIE VERFAHREN DER PSEUDO-BOOLESCHEN PROGRAMMIERUNG

41. Einführung

Die Anwendungsmöglichkeiten der Verfahren der BOOLEschen Programmierung zur Lösung wirtschaftlicher Entscheidungsprobleme sind in zweifacher Hinsicht begrenzt: Die Einschränkungen ergeben sich aus der Notwendigkeit, die Aufgabenstellungen so zu formulieren, daß sie im Entscheidungsmodell durch BOOLEsche Ausdrücke (mit BOOLEschen Verknüpfungen) und ihre Lösungen durch verbal interpretierbare Zeichenkombinationen abgebildet werden können. Letzteres geschieht durch die Bildung der sogenannten Normalformen. Für eine Vielzahl im Grunde zweiwertiger Probleme ist dieses Vorgehen jedoch unmöglich oder ohne Wert, wenn explizite Lösungen erforderlich sind.

Andererseits bleibt die von DANTZIG erstmals klar ausgesprochene Tatsache[1], daß die Lösungen einer Reihe unternehmerischer Entscheidungsprobleme im Hinblick auf die einzelnen in das Modell eingehenden Aktivitäten durch eine Abfolge von Alternationen gebildet werden. Es sind die Fälle, für deren Variable im mathematischen Entscheidungskalkül nur Ja-Nein-Entscheidungen ökonomisch relevant sind. Das Bemühen, hierfür adäquate Lösungsverfahren zu entwickeln, die nicht die Begrenztheit der BOOLEschen Ansätze haben, hat zu der Erarbeitung besonderer, der sogenannten PSEUDO-BOOLEschen Verfahren geführt. Die Arbeit an dem grundlegenden mathematischen Apparat ist noch nicht abgeschlossen, die Untersuchung seiner ökonomischen Möglichkeiten hat gerade erst begonnen, aber bereits bedeutende Fortschritte gemacht. Drei Ansätze sind zu nennen:

---

[1] Vgl. Abschnitt 31. dieser Arbeit.

1. Das von P.L. IVANESCU unter Mitarbeit von S. RUDEANU; I. ROSENBERG und A.DELEANU entwickelte Verfahren[1].

2. Der "Additive Algorithmus" von EGON BALAS[2].

3. Der Ansatz von R. FORTET[3], welcher zeitlich der erste ist.

Die Beiträge 1. und 2. sind aus der Arbeit des Mathematischen Institutes der Rumänischen Akademie der Wissenschaften in Bukarest hervorgegangen. Typisches und für die Berücksichtigung in dieser Arbeit entscheidendes Merkmal der genannten Verfahren ist ihr gemeinsamer Ursprung aus Versuchen, deren Ziel es war, die BOOLEsche Algebra auf ihre Eignung als mathematische Grundlage eigenständiger operationsanalytischer Methoden für spezielle, zweiwertige Entscheidungsprobleme zu prüfen. Da diese Zweiwertigkeit im BOOLEschen Modell implizit berücksichtigt ist und nicht wie bei den herkömmlichen ganzzahligen Programmierungsverfahren zusätzliche Rechenschritte notwendig sind, bieten sich hier besonders für die programmierte Bearbeitung erhebliche rechentechnische Vorteile.

---

[1] Vgl. Ivanescu, P.L.; Rosenberg, I.; Rudeanu, S.: Asupra determinarii minimelor functiilor Pseudobooleenne, in: Studii si cercetari matematice, 14. Jg. 1963, S. 359 ff.
Ivanescu, P.L.; Rosenberg, I.; Rudeanu, S.: Application of Discrete Linear Programming to the minimization of Boolean functions, in: Revue De Mathematiques Pures et Appliquees, 8. Jg. 1963, S. 459 ff. und vor allem Ivanescu, P.L.: Pseudo-Boolean Programming and Applications, a.a.O. Hierin ist ein vollständiges Verzeichnis aller bisherigen Veröffentlichungen von IVANESCU und Mitarbeitern enthalten.

[2] Vgl. Balas, Egon: An Additive Algorithm for Solving Linear Programs with Zero-One Variables, in: JORSA, Vol. 13 No. 4 1965, S. 517 ff.

[3] Vgl. Fortet, R.: L'Algebre de Boole ..., a.a.O., S. 5 ff.
Fortet, R.: Applications de l'Algebre ..., a.a.O., S. 17 ff.

Grundsätzlich kann jedes "außerordentlich ganzzahlige" Programm, in dem als Lösung ein BOOLEscher Vektor mit $(x_1,\ldots,x_n)$ zu bestimmen ist, daß

$$cx' = z$$
$$\text{unter der Bedingung } Ax' \lesseqgtr b',$$

durch Aufzählung und Durchprüfung aller 2^n möglichen 0,1-Zuordnungen gelöst werden. Davon gehen z.B. BELLMAN[1] und BENDERS[2] aus, denn "letzten Endes sind trotz ihres Mangels an Glanz kluge Aufzählungsverfahren in bestimmten Fällen die beste bekannte Lösungsmethode."[3]

Keineswegs müssen in jedem Fall alle 2^n Vektoren tatsächlich durchgeprüft werden. Es lassen sich Kriterien angeben, durch welche bestimmte Zuordnungen ausgeschieden werden können, da sie als optimale Lösungen nicht in Frage kommen[4]. Die Auswahl dieser nicht relevanten Lösungsvektoren ist das typische Merkmal der PSEUDO-BOOLEschen Verfahren. Besonders eindrucksvoll ist dieses Vorgehen am Additiven Algorithmus zu studieren.

---

[1] Vgl. Bellman, Richard: Maximization over Discrete Sets, in Naval Research Logistics Quarterly Vol. 3 1956, S.67 ff.

[2] Vgl. Benders, J.F.: Partitioning Procedures for Solving Mixed Variables Programming Problems, in: Numerische Mathematik 1962, S. 238 ff.

[3] Balinski, M.L.: Integer Programming ..., a.a.O., S. 268: "... clever enumerative schemes - despite their lack of glamour - are in certain situations at least the best known methods of solution."

[3] Vgl. Balinski, M.L.: Integer Programming ..., a.a.O., S.269. Michejew, W.M.: Über Mengen, die eine größte Anzahl paarweise nicht vergleichbarer BOOLEscher Vektoren enthalten, in: Probleme der Kybernetik, Bd. 2 Berlin 1963, S. 75 ff.

42. Das Verfahren von FORTET

Eine einfache, wenn auch sehr spezielle Methode zur Maximalwertbestimmung PSEUDO-BOOLEscher Funktionen ist von FORTET konzipiert worden. Sie stellt kein allgemeines Verfahren dar, wie es der Additive Algorithmus ist, da Beschränkungen nur implizit in der Funktion berücksichtigt werden können.

Jede Funktion

$$f(x_1,\ldots,x_n) \qquad x_i = 0 \text{ oder } 1$$

kann, wenn man für die positive Variable x_i und die negative $\bar{x}_i$ setzt und die dazugehörigen reellen Koeffizienten mit A_i und B_i und konstante Glieder mit C_i bezeichnet, in die Normalform

$$f_i = A_i x_i + B_i \bar{x}_i + C_i \qquad (i=1,\ldots,n)$$

übertragen werden. Für kleinere Programme kann aus dieser Normalform ein rekursives Rechenverfahren wie folgt entwickelt werden: Für jede der n Variablen wird der funktionale Differenzenwert

$$\Delta f_i = f(x_1,\ldots,x_{i-1},x_i,x_{i+1},\ldots,x_n) - f(x_1,\ldots,x_{i-1},\bar{x}_i,x_{i+1},\ldots,x_n)$$

gebildet. Führt man die Subtraktion aus und formt um, ergibt sich:

$$\Delta f_i = (A_i - B_i)(x_i - \bar{x}_i)$$

Diese Beziehung kann als Lösungsbedingung formuliert werden. Eine einfache Überlegung[1] zeigt, daß $f(x_1,\ldots,x_n)$ dann

[1] Wenn die Koeffizientendifferenz einer Variablen x_i positiv ist, muß diese Variable in die Lösung aufgenommen werden (d.h. gleich 1 gesetzt werden), um f zu maximieren, deshalb die Bedingung $(A_i - B_i)(x_i - \bar{x}_i) > 0$. Die strengste Bedingung ist zusätzlich erforderlich, falls $A_i = B_i$ und der Koeffizienten eine weitere BOOLEsche Variable zugeordnet ist. Diese würde sonst unzulässigerweise mit 0 vorherbestimmt.

maximal sein muß, wenn der Differenzenwert für alle i gleich oder größer Null ist.

$$(A_i - B_i)(x_i - \bar{x}_i) \geqq 0$$

Nach Einsetzen der Koeffizienten sind die einzelnen Variablen so zu bestimmen, daß die Bedingung in jedem Fall erfüllt ist. Die optimale Zuordnung ist ermittelt, wenn keine eindeutigen Bedingungen mehr formuliert werden können.

Ein Beispiel soll den theoretischen Ansatz verdeutlichen. Eine Erdölgesellschaft hat 13 Ölfelder (D), zu deren Ausbeutung insgesamt 9 Rohrleitungen (R) verlegt werden müßten. Die Elemente der Inzidenz-Matrix A dieses Streckenkomplexes a_{ij} sind gleich 1, wenn die Rohrleitung R_i zur Erschließung des Feldes D_j notwendig ist, in allen anderen Fällen gleich 0. Die Matrix mit 9 Zeilen und 13 Spalten habe folgendes Aussehen:

$$A = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Die Erträge der Felder sind mit $u_1 = 21$, $u_2 = 16$, $u_3 = 27$, $u_4 = 32$, $u_5 = 12$, $u_6 = 11$, $u_7 = 14$, $u_8 = 5$, $u_9 = 14$, $u_{10} = 26$, $u_{11} = 9$, $u_{12} = 41$, $u_{13} = 13$ und die Aufwendungen für die Rohrleitungen mit $v_1 = 30$, $v_2 = 21$, $v_3 = 25$, $v_4 = 19$, $v_5 = 31$, $v_6 = 15$, $v_7 = 11$, $v_8 = 25$ und $v_9 = 33$ Geldeinheiten angegeben. Es sind die BOOLEschen Variablen

D_j und R_i zu bestimmen, die

$$F = \sum_{j=1}^{13} u_j D_j - \sum_{i=1}^{9} v_i R_i$$

maximieren, wobei die durch A gegebenen Bedingungen, welche als Konjunktionen interpretiert werden können, zu beachten sind:

$D_1 = R_1R_2$; $D_2 = R_2R_3$; $D_3 = R_1R_3R_4$; $D_4 = R_1R_3$; $D_5 = R_4R_8$;

$D_6 = R_2R_5$; $D_7 = R_2R_5$; $D_8 = R_2R_7$; $D_9 = R_4R_7$; $D_{10} = R_4R_7R_9$;

$D_{11} = R_8R_9$; $D_{12} = R_3R_4$; $D_{13} = R_6R_7$;

Ausgeschrieben ergibt sich die PSEUDO-BOOLEsche Funktion

$$F = 21\,R_1R_2 + 16\,R_2R_3 + 27\,R_1R_3R_4 + 32\,R_1R_3 + 12\,R_4R_8$$
$$+ 25\,R_2R_5 + 5\,R_2R_7 + 14\,R_4R_7 + 26\,R_4R_7R_9 + 9\,R_8R_9 + 41\,R_3R_4$$
$$+ 13\,R_6R_7 - 30\,R_1 - 21\,R_2 - 25\,R_3 - 19\,R_4 - 31\,R_5 - 15\,R_6$$
$$- 11\,R_7 - 25\,R_8 - 33\,R_9$$

Als erste Entscheidungsregel wird die Bedingung

$$(1) \quad \Delta f_5 = (25\,R_2 - 31)(R_5 - \overline{R}_5) \geq 0$$

gebildet. Da $25\,R_2 - 31$ immer kleiner 0 ist - für R_2 beliebig 0 oder 1 - muß $R_5 = 0$ gesetzt werden, damit (1) erfüllt ist. Die ursprüngliche Funktion F reduziert sich zu

$$F^{(1)} = 21\,R_1R_2 + 16\,R_2R_3 + 27\,R_1R_3R_4 + 32\,R_1R_3 + 12\,R_4R_8$$
$$+ 5\,R_2R_7 + 14\,R_4R_7 + 26\,R_4R_7R_9 + 9\,R_8R_9 + 41\,R_3R_4$$
$$+ 13\,R_6R_7 - 30\,R_1 - 21\,R_2 - 25\,R_3 - 19\,R_4 - 15\,R_6$$
$$- 11\,R_7 - 25\,R_8 - 33\,R_9$$

$$(2)\quad \Delta f_6^{(1)} = (13\,R_7 - 15)(R_6 - \overline{R}_6) \geqq 0$$

Aus $13\,R_4 - 15 < 0 \curvearrowright R_6 = 0$

$$F^{(2)} = 21\,R_1R_2 + 16\,R_2R_3 + 27\,R_1R_3R_4 + 32\,R_1R_3 + 12\,R_4R_8$$
$$+ 5\,R_2R_7 + 14\,R_4R_7 + 26\,R_4R_7R_9 + 9\,R_8R_9 + 41\,R_3R_4$$
$$- 30\,R_1 - 21\,R_2 - 25\,R_3 - 19\,R_4 - 11\,R_7 - 25\,R_8 - 33\,R_9$$

$$(3)\quad \Delta f_8^{(2)} = (12\,R_4 + 9\,R_9 - 25)(R_8 - \overline{R}_8) \geqq 0$$

Aus $12\,R_4 + 9\,R_9 - 25 < 0 \curvearrowright R_8 = 0$

$$F^{(3)} = 21\,R_1R_2 + 16\,R_2R_3 + 27\,R_1R_3R_4 + 32\,R_1R_3 + 5\,R_2R_7$$
$$+ 14\,R_4R_7 + 26\,R_4R_7R_9 + 41\,R_3R_4 - 30\,R_1 - 21\,R_2 - 25\,R_3$$
$$- 19\,R_4 - 11\,R_7 - 33\,R_9$$

$$(4)\quad \Delta f_9^{(3)} = (26\,R_4R_7 - 33)(R_9 - \overline{R}_9) \geqq 0$$

Aus $26\,R_4R_7 - 33 < 0 \curvearrowright R_9 = 0$

$$F^{(4)} = 21\,R_1R_2 + 16\,R_2R_3 + 27\,R_1R_3R_4 + 32\,R_1R_3 + 5\,R_2R_7$$
$$+ 14\,R_4R_7 + 41\,R_3R_4 - 30\,R_1 - 21\,R_2 - 25\,R_3 - 19\,R_4$$
$$- 11\,R_7$$

Da weitere Bedingungen nicht gefunden werden können, ist durch $F^{(4)}$ mit $R_1 = R_2 = R_3 = R_4 = R_7 = 1$ und $R_5 = R_6 = R_8 = R_9 = 0$ der maximale Wert $F_{max} = 50$ GE ermittelt. Unter den $2^9 = 512$ möglichen BOOLEschen Vektoren ist mit (1, 1, 1, 1, 0, 0, 1, 0, 0) die optimale Zuordnung fixiert worden, d.h. im optimalen Programm werden die Felder 1, 2, 3, 4, 8, 9 und 12 mit Hilfe der Rohrleitungen

1, 2, 3, 4 und 7 erschlossen. Für kleinere Programme, die in die dargestellte Form gebracht, d.h. in einem komplexen Ausdruck zusammengefaßt werden können, führt die Methode von FORTET schnell zum Ziel, rechentechnische Schwierigkeiten ergeben sich nicht.

43. IVANESCUs PSEUDO-BOOLEsche Programmierung

431. Das Verfahren

Der grundlegende und richtungsweisende Ansatz für die von ihm erstmals so definierte PSEUDO-BOOLEsche Programmierung stammt von IVANESCU. Sein Vorgehen ist mathematisch beachtenswert, hat sich als rechentechnisch sehr wirkungsvoll erwiesen und ist leicht programmierbar. Darüber hinaus zeigt die Vielfalt der mit dem Verfahren erfolgreich in Angriff genommenen Aufgabenstellungen[1] seine umfassenden Anwendungsmöglichkeiten.

Bezeichnet man mit IVANESCU die BOOLEsche Algebra über die Menge $\{0,1\}$ mit L_2 und die Menge der geordneten Paare bei n Variablen mit $L_2{}^{n2}$, d.h.

$$L_2{}^n = \left\{(c_1,\ldots,c_n) \,\middle|\, c_i \in L_2,\ (i=1,\ldots,n)\right\},$$

dann stellt sich die PSEUDO-BOOLEsche Funktion F (PBF) als Abbildung von $L_2{}^n$ in die Menge der reellen oder für Sonderfälle der ganzen Zahlen dar[3]:

---

[1] Vgl. Ivanescu, P.L.; Rosenberg, I.: Application of Pseudo-Boolean Programming ..., a.a.O., S. 163.
Ivanescu, P.L.: Some Network Flow Problems ..., a.a.O., S. 388 ff.
Ivanescu, P.L.; Rudeanu, S.: A Pseudo-Boolean Approach to ..., a.a.O.,
Ivanescu, P.L.: Pseudo-Boolean Programming ..., a.a.O.

[2] Entspricht dem kartesischen Produkt von Mengen, hier also $L_2 x L_2 x L_2 x \ldots x L_2$.

[3] Vgl. hierzu die Definitionen in Abschnitt 33.

$$F : L_2^n \longrightarrow \mathbb{R} \quad (\triangleq F(x_1,\ldots,x_n) : L_2 \times L_2 \times \ldots \times L_2 = L_2^n \longrightarrow \mathbb{R})$$

oder $F : L_2^n \longrightarrow \mathbb{Z}$

Das Verfahren besteht darin, Vektoren der Art

$$(x_1,\ldots x_n) \in L_2^n$$

zu ermitteln, welche $F(x_1,\ldots,x_n)$ minimieren. Die von uns angedeutete Schwierigkeit der Transformation boolescher in arithmetische Verknüpfungen[1] innerhalb des Rechenganges überwindet IVANESCU durch

$$a \vee b = a + b - ab^2 \qquad (1)$$

und $$\bar{a} = 1 - a \qquad (2)$$

Entsprechend gilt für mehrere Variable :

$$a \vee b \vee c = a + b + c - ab - bc - ca + abc$$

sowie $$a \vee b \vee c \vee d = a + b + c + d - ab - bc - cd - da + abc + bcd + cda + dab - abcd,$$

worin + und - als gewöhnliche arithmetische Operationen zu interpretieren sind.

Der Wert einer PBF bleibt offenbar unverändert, wenn eine einzelne Variable mit ihren möglichen Werten explizit eingeführt wird[3], d.h.:

---

[1] Vgl. Abschnitt 31.

[2] Für die BOOLEsche Addition (logische Disjunktion) setzen wir hier in Abweichung zum dritten Kapitel zur besseren Unterscheidung das Symbol "v", sonst "+".

[3] Vgl. Ivanescu, P.L.; Rosenberg, I.; Rudeanu, S.: Asupra Determinarii minimelor ..., a.a.O., S. 359 ff. Ivanescu, P.L.: Application of Pseudo-Boolean ..., a.a.O., S. 165

$$F^{(1)}(x_1,\ldots,x_n)=x_1F^{(1)}(1,x_2,\ldots,x_n)+\bar{x}_1F^{(1)}(0,x_2,\ldots,x_n) \quad (3)$$

Wegen (2) ergibt sich

$$\begin{aligned}F^{(1)}(x_1,\ldots,x_n)&=x_1F^{(1)}(1,x_2,\ldots,x_n)+(1-x_1)F^{(1)}(0,x_2,\ldots,x_n)\\&=x_1\left[F^{(1)}(1,x_2,\ldots,x_n)-F^{(1)}(0,x_2,\ldots,x_n)\right]\\&\quad+F^{(1)}(0,x_2,\ldots,x_n)\end{aligned}$$

Setzt man weiter

$$\begin{aligned}F^{(1)}(1,x_2,\ldots,x_n)-F^{(1)}(0,x_2,\ldots,x_n) &= g_1 \quad \text{und}\\ F^{(1)}(0,x_2,\ldots,x_n) &= h_1\end{aligned} \quad (3.1)$$

so kann (3) in der Form $F^{(1)}=g_1x_1+h_1$ geschrieben werden, wobei g_1 und h_1 PBF von $x_2,\ldots,x_n$ sind. Dehnt man diese Überlegung auf alle Variablen aus, folgt induktiv, daß jede PBF als Polynom mit reellen (oder ganzen) Koeffizienten darstellbar ist, linear in jeder Variablen[1]. Der Rechengang besteht aus:

1. Entsprechend (3) und (3.1)wird gebildet

$$F^{(1)}(x_1,\ldots,x_n)=x_1g_1(x_2,\ldots,x_n)+h_1(x_2,\ldots,x_n) \quad (4)$$

Hierfür bestimmt man die Wertzuordnungen a_i bzw. $b_i \in \{0,1\}$, für welche die angegebenen Bedingungen gelten:

$$M_1 = \left\{(a_2,\ldots,a_n) \in L_2^{n-1} \,\middle|\, g_1(a_2,\ldots,a_n) < 0\right\} \quad (4.1)$$

---

[1] Als 1. Theorem der Pseudo-Booleschen Programmierung formuliert,
vgl. Ivanescu, P.L.: Pseudo-Boolean Programming ..., a.a.O., S. 5. Zum Beweis vgl.
Ivanescu, P.L.; Rosenberg, I.; Rudeanu, S.: Asupra Determinarii ..., a.a.O., S. 359 ff.

oder

$$N_1 = \left\{ (b_2,\ldots,b_n) \in L_2^{n-1} \;\middle|\; g_1(b_2,\ldots,b_n) = 0 \right\} \qquad (4.2)$$

Für diese Wertzuordnungen ermitteln wir x_1 als disjunktive Verknüpfung

$$x_1 = \bigsqcup_{(a_2,\ldots,a_n) \in M_1} x_2^{a_2} \ldots x_n^{a_n} \vee u_1 \left[\bigsqcup_{(b_2,\ldots,b_n) \in N_1} x_2^{b_2} \ldots x_2^{b_n} \right] \qquad (4.3)$$

worin u_1 ein beliebiger Wert aus L_2 ist[1].

x_1' sei der Wert von x_1, wenn $u_1 = 0$ gesetzt wird, d.h.

$$x_1' = \bigsqcup_{(a_2,\ldots,a_n) \in M_1} x_2^{d_2} \ldots x_n^{a_n} \qquad (4.4)$$

Das arithmetische Äquivalent x_1^* von x_1' erhält man, indem

$$x_i^{a_i} = \begin{cases} x_i, & \text{wenn } a_i = 1 \\ \overline{x_i}, & \text{wenn } a_i = 0 \end{cases}$$

gesetzt wird durch wiederholte Anwendung der Formeln (1) und (2):

$$x_1^* = \bigvee_{(a_2,\ldots,a_n) \in M_1} x_2^{a_2} \ldots x_n^{a_n}$$

---

1 Wenn man nur ein Minimum von F ermitteln will, genügt das Weiterrechnen allein mit den Wertzuordnungen aus M_1, u_1 wird dann mit 0 angesetzt (deshalb die eckige Klammer um den 2. Teil des Ausdrucks). Sollen alle Minima angegeben werden, müssen auch alle Werte berücksichtigt werden, welche die für $N_1,\ldots,N_n$ geltende Bedingung erfüllen, d.h. u_i muß jeweils mit 0 und 1 eingeführt werden. Hier besteht eine gewisse Parallele zu FORTET.

Man hat $x_1 = x_1^*(x_2,\ldots,x_n)$

2. Nun bildet man

$F^{(2)}(x_2,\ldots,x_n)$, indem in $F^{(1)}$ x_1 durch $x_1^*(x_2,\ldots,x_n)$ ersetzt wird:

$$F^{(2)}(x_2,\ldots,x_n) = F^{(1)}\left[x_1^*(x_2,\ldots,x_n), x_2,\ldots,x_n\right]$$

$$= x_2 g_2(x_3,\ldots,x_n) + h_2(x_3,\ldots,x_n)$$

Wieder bildet man die Mengen M und N;

$$M_2 = \left\{(a_3,\ldots,a_n) \in L_2^{n-2} \,\middle|\, g_2(a_3,\ldots,a_n) < 0\right\}$$

$$N_2 = \left\{(b_3,\ldots,b_n) \in L_2^{n-2} \,\middle|\, g_2(b_3,\ldots,b_n) = 0\right\}$$

und daraus

$$x_2 = \bigsqcup_{(a_3,\ldots,a_n)\in M_2} x_3^{a_3}\ldots x_n^{a_n} \vee u_2 \left[\bigsqcup_{(b_3,\ldots,b_n)\in N_2} x_3^{b_3}\ldots x_n^{b_n}\right.$$

sowie

$$x_2^* = \bigvee_{(a_3,\ldots,a_n)\in M_2} x_3^{a_3}\ldots x_n^{a_n}$$

und verfährt weiter wie unter 1.

Allgemein gilt für die einzelnen Rechenschritte folgende Vorschrift:

Bilde

$$F^{(i)}(x_i,\ldots,x_n) = F^{(i-1)}\left[x_{i-1}^*(x_i,\ldots,x_n),\ x_i,\ldots,x_n\right]$$

$$= x_i g_i(x_{i+1},\ldots,x_n) + h_i(x_{i+1},\ldots,x_n)$$

Bestimme

$$M_i = \left\{(a_{i+1},\ldots,a_n) \in L_2^{n-i} \;\middle|\; g_i(a_{i+1},\ldots,a_n) < 0\right\} \quad \text{und}$$

$$N_i = \left\{(b_{i+1},\ldots,b_n) \in L_2^{n-i} \;\middle|\; g_i(b_{i+1},\ldots,b_n) = 0\right\}$$

Errechne

$$x_i = \bigsqcup_{(a_{i+1},\ldots,a_n)\in M_i} x_{i+1}^{a_{i+1}}\ldots x_n^{a_n} \;\vee\; u_i\left[\bigsqcup_{(b_{i+1},\ldots,b_n)\in N_i} x_{i+1}^{b_{i+1}}\ldots x_n^{b_n}\right]$$

sowie

$$x_i^* = \bigvee_{(a_{i+1},\ldots,a_n)\in M_i} x_{i+1}^{a_{i+1}}\ldots x_n^{a_n} \quad \text{mit } u_i = 0\,[\text{oder } 1]$$

Schließlich verbleibt $F_n(x_n) = x_n g_n + h_n$ (5)
worin g_n und h_n Konstante sind.

Für (5) wird gesetzt

$$x_n = \begin{cases} 1, & \text{wenn } g_n < 0 \\ 0, & \text{wenn } g_n > 0 \\ u_n, & \text{wenn } g_n = 0 \end{cases} \qquad (6)$$

wobei u_n wieder beliebig 0 oder 1 ist. Aus (6) ergibt sich, daß x_n schließlich eine Funktion von u_n ist.

$$x_n = X_n(u_n)$$

Hieran schließt sich die rekursive, explizite Bestimmung von x_{n-1} bis x_1 an. Aus (6) und (7) folgt durch Einsetzen

$$\begin{aligned} x_{n-1} &= X_{n-1}(u_{n-1}, x_n) \\ &= X_{n-1}\left[u_{n-1}, X_n(u_n)\right] \\ &= X_{n-1}(u_{n-1}, u_n) \end{aligned}$$

und $x_{n-2} = X_{n-2}(u_{n-2}, u_{n-1}, u_n)$

usw. bis $x_1 = X_1(u_1, u_2, \ldots, u_n)$

Allgemein ist $x_i = X_i(u_i, u_{i+1}, \ldots, u_n)$

IVANESCU hat nachgewiesen, daß für jeden auf diese Weise ermittelten BOOLEschen Vektor die PBF (4) einen minimalen Wert annimmt[1]. Besonders wirkungsvoll ist der dargestellte Algorithmus, wenn man mit der Bestimmung der Variablen x_i einer Funktion F_j beginnt, die von allen Variablen in den Gliedern der Funktion am wenigsten enthalten ist[2].

432. Erläuterung des Algorithmus an einem numerischen Beispiel

Gegeben sei aus L_2^5 die Funktion:

$$F^{(1)}(x_1, \ldots, x_5) = 3x_1x_2 - 5x_3x_5 + 2x_2x_3 - 4x_5 + 3x_4x_5 + 9x_4 - 5x_1x_3 - x_3x_4$$

Unter Beachtung von (3) wird gebildet

$$g_1(x_2, .., x_5) = F^{(1)}(1, x_2, .., x_5) - F^{(1)}(0, x_2, .., x_5)$$

$$= 3x_2 - 5x_3$$

Daraus ergibt sich entsprechend (4.1;4.2)

$$M_1 = \{(1,1), (0,1)\} \qquad \text{und } N_1 = \{(0,0)\}$$

---

[1] Vgl. zum Beweis: Ivanescu, P.L.; Rosenberg, I.; Rudeanu, S.: Asupra determinarii ..., a.a.O., S. 361 f.

[2] Vgl. Ivanescu, P.L.: Pseudo-Boolean Programming ..., a.a.O., S. 11, Anmerkung 2

Durch Anwendung von (4.3) erhält man dann

$$x_1 = x_2^{1}x_3^{1} \vee x_2^{0}x_3^{1} \vee u_1(x_2^{0}x_3^{0})$$

sowie $x_1' = x_2x_3 \vee \bar{x}_2x_3$ für $u_1 = 0$ (vgl. 4.4)

was über $x_1^* = x_2x_3 + \bar{x}_2x_3 - x_2x_3\bar{x}_2x_3$ wegen $\bar{x}_i = 1-x_i$

zu $x_1^* = x_3$ wird[1].

Nach Einsetzen in $F^{(1)}$ ergibt sich

$$F^{(2)}(x_2,..,x_5) = 3x_2x_3 - 5x_3x_5 + 2x_2x_3 - 4x_5 + 3x_4x_5 + 9x_4 - 5x_3 - x_3x_4$$

$$= 5x_2x_3 - 5x_3x_5 - 4x_5 + 3x_4x_5 + 9x_4 - 5x_3 - x_3x_4$$

Für g_2 gilt dann $g_2(x_3,..,x_5) = 5x_3$ mit $M_2 = \emptyset$[2] und $N_2 = \{(0)\}$, so daß $x_2 = u_2 \cdot x_3^0$ und $x_2^* = x_2^* = 0$.

Damit erhält man

$$F^{(3)}(x_3,..,x_5) = -5x_3x_5 - 4x_5 + 3x_4x_5 + 9x_4 - 5x_3 - x_3x_4$$

mit $g_3(x_4,x_5) = -x_4 - 5x_5 - 5$

wodurch $M_3 = \{(1,1),(1,0),(0,1)\}$ und $N_3 = \emptyset$ fixiert sind.

Aus $x_3 = x_3' = x_4^1x_5^1 \vee x_4^1x_5^0 \vee x_4^0x_5^1 = x_4x_5 \vee x_4\bar{x}_5 \vee \bar{x}_4x_5$

$$= x_4 \vee x_5$$

---

[1] Selbstverständlich ist auch im BOOLEschen System $x_1^0 = x_2x_3 \vee \bar{x}_2x_3 = x_3$, da $x_i \vee \bar{x}_i = 1$ und $x_i \wedge \bar{x}_i = 0$ und $x_i \wedge x_i = x_i^1$ ist. Vgl. hierzu die Axiome der BOOLEschen Algebra im 1. Kapitel.

[2] $\emptyset$ symbolisiert die leere Menge.

folgt $x_3^* = x_4 + x_5 - x_4x_5$

$F^{(4)}$ ist somit gegeben durch

$$F^{(4)}(x_4,x_5) = -5x_4x_5-5x_5+5x_4x_5-4x_5-3x_4x_5+9x_4-5x_4-5x_5+5x_4x_5$$

$$-x_4-x_4x_5+x_4x_5 = -14x_5+3x_4+3x_4x_5$$

und $g_4(x_5) = 3+3x_5$ mit $M_4 = \emptyset$ und $N_4 = \emptyset$.

Damit ist $x_4 = x_4' = x_4^* = 0$ fixiert.

Nach dem Einsetzen in $F^{(4)}$ verbleibt $F^{(5)}(x_5) = -14x_5$ und $g_5 = -14$

Nach (6) muß $x_5 = 1$ gesetzt werden, da $g_5 < 0$.

Die einzelnen PSEUDO-BOOLEschen Funktionen können nun wie folgt rekursiv aufgelöst werden: Aus

$$x_5 = 1 \curvearrowright g_4 > 0 \curvearrowright x_4 = 0$$
$$\curvearrowright g_3 < 0 \curvearrowright x_3 = 1$$
$$\curvearrowright g_2 > 0 \curvearrowright x_2 = 0$$
$$\curvearrowright g_1 < 0 \curvearrowright x_1 = 1$$

Der ermittelte Lösungsvektor ist demnach mit (1,0,1,0,1) fixiert, der entsprechende Funktionswert ist $F_{min} = -14$. F hat ein einziges absolutes Minimum, jede andere Zuordnung führt zu einem ungünstigerem Wert.

Es wurde festgestellt, daß mit einiger Übung auch komplizierte PSEUDO-BOOLEsche Funktionen schnell gelöst werden können. Wichtig ist, daß mehrere Minima um so häufiger auftreten je umfangreicher die Ausdrücke sind. Auf keinen Fall dürfen dann die Elemente aus N_i unberücksichtigt bleiben. Für die Funktion $F_1(x_1,\ldots,x_5) = 2x_1+3x_2-7x_3-5x_1x_2x_3+3x_2x_4+9x_4x_5$ [1] sind

---

[1] Vgl. Ivanescu, P.L.; Rosenberg, I.; Rudeanu, S.: Asupra determinarii ..., a.a.O., S. 363.

z.B. folgende fünf Vektoren gleichwertig:

$$F_1(0,0,1,0,0) = F_1(0,0,1,0,1) = F_1(1,1,1,0,0) = F_1(1,1,1,0,1)$$

$$= F_1(0,0,1,1,0) = -7$$

Besonders für unternehmerische Entscheidungssituationen ist es oft wertvoll, mehrere gleichwertige Alternativen zu kennen. Bei der Auswahl aus einem derartig homogenen Kollektiv können dann zusätzlich - praktisch in einem zweiten Schritt - nicht quantivizierbare Präferenzen berücksichtigt werden: z.B. persönliche Neigungen des unternehmerischen Entscheidungsträgers wie die Sympathie für einen bestimmten Lieferanten oder Kunden. Derartige Unwägbarkeiten wurden bislang aus der Entscheidungstheorie ausgeklammert, oder im Hinblick auf das Rationalprinzip unter eine komplexe Zielfunktion subsumiert. (Nutzenschätzungen) In eine solche Zielfunktion würde als Nutzenzugang der positive Einfluß einer aus Sympathie getroffenen Entscheidung eingehen.

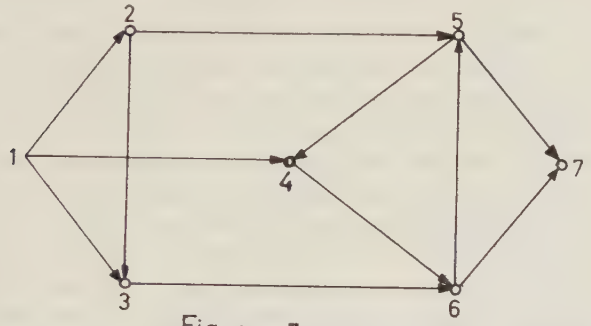
433. Die Anwendung auf Netzwerk-Fluß-Probleme als Beispiel für die ökonomische Relevanz der PSEUDO-BOOLEschen Programmierung

Wie bereits angedeutet soll die Leistungsfähigkeit der PSEUDO-BOOLEschen Programmierung am klassischen Transportnetzmodell durch Bestimmung des maximalen Flusses gezeigt werden. Als Ausgangspunkt dienen dabei die grundlegenden Arbeiten und Ergebnisse von FORD und FULKERSON[1].

---

[1] Vgl. Ford, L.R.; Fulkerson, D.R.: Flows in Networks, Princeton 1962.
Fulkerson, D.R.: Flows in Networks, in: Recent Advances in Mathematical Programming, hrsgg. von Graves, R.L. und Wolfe, P., New - York - Toronto - London 1963, S. 319 ff.
Hadley, G.: Linear Programming, Reading - Palo Alto - London 1963, S. 331 ff.

Gegeben sei ein Netzwerk, dem der gerichtete endliche Graph $G = [N,A]$ aus Figur 7 zugrunde liegt.



Figur 7

Damit ist die nebenstehende Inzidenz-Matrix

$$A = \|a_{ij}\|$$

gegeben, für die gilt:

$$a_{ij} = \begin{cases} 1, & \text{wenn } (x_i,x_j) \in A \\ 0, & \text{wenn } (x_i,x_j) \notin A^1. \end{cases}$$

a_{ij}	1	2	3	4	5	6	7
1	0	1	1	1	0	0	0
2	0	0	1	0	1	0	0
3	0	0	0	0	0	1	0
4	0	0	0	0	0	1	0
5	0	0	0	1	0	0	1
6	0	0	0	0	1	0	1
7	0	0	0	0	0	0	0

x_1 und x_7 sind hierbei als Ursprungs- bzw. Mündungspunkt eines Stroms durch das Netzwerk angenommen. Die einzelnen Kanten haben bestimmte Kapazitäten c_{ij}, wobei die Bedingung

$$a_{ij} = 0 \Longrightarrow c_{ij} = 0$$

zu beachten ist.

---

[1] Die Anordnung der Knoten entspricht der Richtung ihrer gemeinsamen Kante.

Die Matrix $C = \|c_{ij}\|$ der Kanten-Kapazitäten[1] ist

c_{ij}	1	2	3	4	5	6	7
1	0	3	4	9	0	0	0
2	0	0	5	0	2	0	0
3	0	0	0	0	0	6	0
4	0	0	0	0	0	8	0
5	0	0	0	4	0	0	7
6	0	0	0	0	1	0	9
7	0	0	0	0	0	0	0

Für die einzelnen Teilströme innerhalb der Kanten des Netzwerkes, welche durch die Matrix $S = \|s_{ij}\|$ abgebildet werden, gelten somit folgende Beschränkungen:

$$s_{ij} \leqq c_{ij}$$

$$\text{und} \sum_{j=1}^{n} a_{ij}s_{ij} - \sum_{j=1}^{n} a_{ji}s_{ji} = \begin{cases} w, \text{ wenn } i = 1^{2} \\ 0, \text{ wenn } i \neq 1,n \\ -w, \text{ wenn } i = n \end{cases}$$

w ist dabei gleich dem Wert des Gesamtstromes, welcher unter Beachtung dieser Bedingungen von x_1 nach x_7 fließen kann[3]. Die Lösung des Problems fordert die Bestimmung des maximalen Flußwertes für das Netz und die Teilströme, aus denen sich der maximale Fluß zusammensetzt. Das PSEUDO-BOOLEsche Programm

---

[1] Der Einfachheit halber sind hier ganzzahlige Werte für die Kapazitäten angenommen. Das bedeutet aber keine Einschränkung der Allgemeingültigkeit: Reelle Werte könnten nämlich mit einer für praktische Zwecke ausreichenden Genauigkeit durch rationale Werte angenähert, diese durch die Multiplikation mit einem eindeutig bestimmbaren Hauptnenner n (eindeutig, da endlich viele Kapazitäten) in ganzzahlige Werte umgeformt werden. Um die tatsächlichen Kapazitäten zu erhalten, teilt man die Lösungswerte wieder durch n.

[2] Aus dieser Bedingung geht hervor, daß die Sickerquote gleich 0 ist. Der Wert des Flusses muß für alle Zwischenpunkte gleich 0 sein, da der Wert der ankommenden gleich dem Wert der abgehenden Teilströme ist.

[3] Vgl. Ford, L.R.; Fulkerson, D.R.: Flows in Networks, a.a.O., S. 4.

für diese Aufgabe basiert auf einigen grundlegenden Sätzen über Graphen und Netzwerke:

Damit überhaupt ein positiver Fluß von x_1 nach x_n fließen kann, ist eine Abfolge von gerichteten Kanten der Art $\{(x_1,x_2),(x_2,x_3),\ldots,(x_{i-1},x_i),(x_i,x_{i+1}),\ldots,(x_{n-2},x_{n-1}),(x_{n-1},x_n)\}$ notwendig, die jeweils positive Kapazitäten aufweisen, so daß der Bestimmungsort x_n durch Wandern auf einer Kette[1] von Kanten entlang ihrer Richtung von x_1 aus erreicht werden kann. Teilt man die Menge aller Knoten des Netzwerkes in zwei Teilmengen K und $\overline{K}$ - wobei $\overline{K} = N - K$ ist - derart auf, daß der Ursprungspunkt x_1 ein Element der einen Teilmenge ($x_1 \in K$) und der Bestimmungspunkt Element der anderen Menge ist ($x_n \in \overline{K}$), so wird durch jede Teilmenge K_i ein Schnitt durch das Netzwerk festgelegt. Er besteht offenbar aus der Menge gerichteter Kanten, die von jeder möglichen Kette - welche von x_1 nach x_n führt - mindestens eine Kante enthält. Von dieser Kante gehört der eine Knoten zur Teilmenge K, der andere zu $\overline{K}$. Die Kapazität dieser Kante, welche durchschnitten ist, wird mit c (x_i,x_j) bezeichnet, wobei $x_i \in K, x_j \in \overline{K}$ ist[2]. Die gesamte Kapazität C $(K,\overline{K})$ des Schnittes (Schnittwert) ist gleich der Summe aller positiven Kapazitäten der auf dem Schnitt liegenden Kanten. Der maximale Fluß durch ein Netzwerk ist offenbar erreicht, wenn die kapazitativen Möglichkeiten des Netzes vollkommen ausgeschöpft sind, d. h. wenn die positiven Verbindungen von Punkten in K zu Punkten in $\overline{K}$ voll ausgelastet sind[3].

---

[1] Eine Kette enthält nur gleichgerichtete, Wege dagegen auch entgegengerichtete Kanten.
Vgl. Ford, L.R.; Fulkerson, D.R.: Flows in Networks, a.a.O., S. 3. Allerdings ist die Terminologie nicht immer einheitlich, vgl. deshalb auch Dantzig, G.B.: Lineare Programmierung und Erweiterungen, a.a.O., S. 461.

[2] Mit anderen Worten: x_i ist abgebende, $\bar{x}_j$ empfangende Stelle, für $x_i \longrightarrow x_j$ steht $^1(x_i,\bar{x}_j)$.

[3] Dantzig, spricht von saturierten Kanten.
Vgl. Dantzig, G.B.: Lineare Programmierung und Erweiterungen a.a.O., S. 447.

FORD und FULKERSON haben nachgewiesen, daß dieser maximale Flußwert gleich dem Wert des "minimalen Schnittes" ist[1]:

$$w_{max} = C(K_0, \overline{K}_0)$$

Auf dieser Feststellung baut das PSEUDO-BOOLEsche Programm auf; der maximale Fluß wird mit Hilfe des minimalen Schnittes errechnet. Den minimalen Schnitt erhält man, wenn die durch Abbildung von A in C erhaltene Kapazitätsfunktion des Netzwerkes minimiert wird. Der Schnittwert in Abhängigkeit von der durch K und $\overline{K}$ angegebenen Trennung des Netzes ist

$$C(K,\overline{K}) = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_i \overline{x}_j$$

Dieser Ausdruck kann als PSEUDO-BOOLEsche Funktion definiert werden. Die Koeffizienten c_{ij} sind der Matrix C zu entnehmen, für die Variablen treffen wir folgende Vereinbarung: Offenbar kann man jede Teilmenge K von N ($K \subset N$; $N = \{x_1, \ldots, x_n\}$) dadurch kennzeichnen, daß für sie ein Vektor $(x_1, \ldots, x_n)$ gebildet wird, indem man für x_i

$$x_i = \begin{cases} 1, \text{ wenn } x_i \in K \\ 0, \text{ wenn } x_i \notin K \end{cases}$$

setzt. Damit hat jeder Schnitt die Form

$$(1, x_2, \ldots, x_{n-1}, 0), \text{ weil } x_1 \in K \text{ und } x_n \in K.$$

Als Lösung ist also wiederum ein BOOLEscher Vektor $(1, x_2^0, \ldots, x_{n-1}^0, 0)$ zu bestimmen, welcher die PBF minimiert.

---

[1] Vgl. Ford, L.R.; Fulkerson, D.R.: Maximal Flow through ..., a.a.O., S. 399 ff.
Ford, L.R.; Fulkerson, D.R.: Flows in Networks, a.a.O., S. 11 ff.

Der minimale Schnitt ist durch die Menge aller Knoten gegeben, die mit 1 errechnet werden[1].

Für das obige Beispiel erhält man unter Beachtung, daß $x_1^0 = 1$ und $x_7^0 = 0$, d.h. $\bar{x}_7^0 = 1$ die PSEUDO-BOOLEsche Funktion

$$C_2(x_2,\ldots,x_6) = 3\bar{x}_2 + 4\bar{x}_3 + 9\bar{x}_4 + 5x_2\bar{x}_3 + 2x_2\bar{x}_5 + 6x_3\bar{x}_6 + 8x_4\bar{x}_6 + 4x_5\bar{x}_4 + 7x_5 + x_6\bar{x}_5 + 9x_6$$

Das ist wegen $\bar{x}_i = 1 - x_i$

$$= 3 - 3x_2 + 4 - 4x_3 + 9 - 9x_4 + 5x_2 - 5x_2x_3 + 2x_2 - 2x_2x_5 + 6x_3$$
$$- 6x_3x_6 + 8x_4 - 8x_4x_6 + 4x_5 - 4x_4x_5 + 7x_5 + x_6 - x_5x_6 + 9x_6$$

$$C_2(x_2,\ldots,x_6) = 16 + 4x_2 + 2x_3 - x_4 + 11x_5 + 10x_6 - 5x_2x_3 - 2x_2x_5 - 6x_3x_6$$
$$- 8x_4x_6 - 4x_4x_5 - x_5x_6$$

Verfährt man jetzt nach den oben dargestellten Lösungsverfahren, ergibt sich:

1. $g_2(x_3,..,x_6) = 4 - 5x_3 - 2x_5$; $M_2 = \{(1,1), (1,0)\}$; $N_2 = \emptyset$

   Dann ist $x_2 = x_3^1x_5^1 \vee x_3^1x_5^0$ und $x_2 = x_3x_5 \vee x_3\bar{x}_5$

   das wird zu $x_2^* = x_3$ wegen $x_i \vee \bar{x}_i = 1$

   Nach Einsetzen und Kürzen hat man

2. $C_3(x_3,..,x_6) = 16 + x_3 - x_4 + 11x_5 + 10x_6 - 2x_3x_5 - 6x_3x_6 - 8x_4x_6 - 4x_4x_5 - x_5x_6$

   und $g_3(x_4,..,x_6) = 1 - 2x_5 - 6x_6$

   mit $M_3 = \{(1,1),(1,0),(0,1)\}$ und $N_3 = \emptyset$

---

[1] Andere Lösungsmethoden sind z.B. die sogenannte Baum-Methode oder die Kennzeichnungsmethode,
vgl. dazu Hadley, G.: Linear Programming, a.a.O., S. 337 ff. und Dantzig, G.B.: Lineare Programmierung und Erweiterungen, a.a.O., S. 452 ff.

So erhält man $x_3 = x_5^1 x_6^1 \vee x_5^1 x_6^0 \vee x_5^0 x_6^1$

und $x_3' = x_5 x_6 \vee x_5 \bar{x}_6 \vee \bar{x}_5 x_6$

Danach ist $x_3^* = x_5 \vee x_6 = x_5 + x_6 - x_5 x_6$ und man hat

3. $C_4(x_4,..,x_6) = 16+x_5+x_6-x_5x_6-x_4+11x_5+10x_6-2x_5-2x_5x_6+2x_5x_6$

$-6x_5x_6-6x_6+6x_5x_6-8x_4x_6-4x_4x_5-x_5x_6$

$= 16-x_4+10x_5+5x_6-2x_5x_6-8x_4x_6-4x_4x_5$

Somit wird $g_4(x_5,x_6) = -1-4x_5-8x_6$

mit $M_4 = \{(1,1), (1,0), (0,1), (0,0)\}$ und $N_4 = \emptyset$

$x_4 = x_5^1 x_6^1 \vee x_5^1 x_6^0 \vee x_5^0 x_6^1 \vee x_5^0 x_6^0$

$x_4' = x_5 x_6 \vee x_5 \bar{x}_6 \vee \bar{x}_5 x_6 \vee \bar{x}_5 \bar{x}_6$

$x_4^* = 1$

Damit verbleibt nach dem Einsetzen in C_4

4. $C_5(x_5,x_6) = 15+6x_5-3x_6-2x_5x_6$

und $g_5(x_6) = 6-2x_6$ mit $M_5 = \emptyset$ und $N_5 = \emptyset$,

dann ist $x_5 = x_5' = x_5^* = 0$ und man hat

5. $C_6(x_6) = 15-3x_6$

$g_6 = -3$

Daraus folgt wegen (6) $x_6 = \bar{x}_6 = x_6^* = x_6^0 = 1$

Mit der Bestimmung von $x_6^0 = 1$ als erste Lösungskomponente ist der Algorithmus abgeschlossen und durch rekursives Lösen der g-Funktionen bei Beachtung von (6) werden die übrigen Lösungswerte ermittelt.

Aus $x_6^0 = 1$ folgt $g_5 > 0$ und daraus $x_5^0 = 0$,

aus $x_6^0 = 1$ und $x_5^0 = 0$ $\curvearrowright$ $g_4 < 0$ $\curvearrowright$ $x_4^0 = 1$,

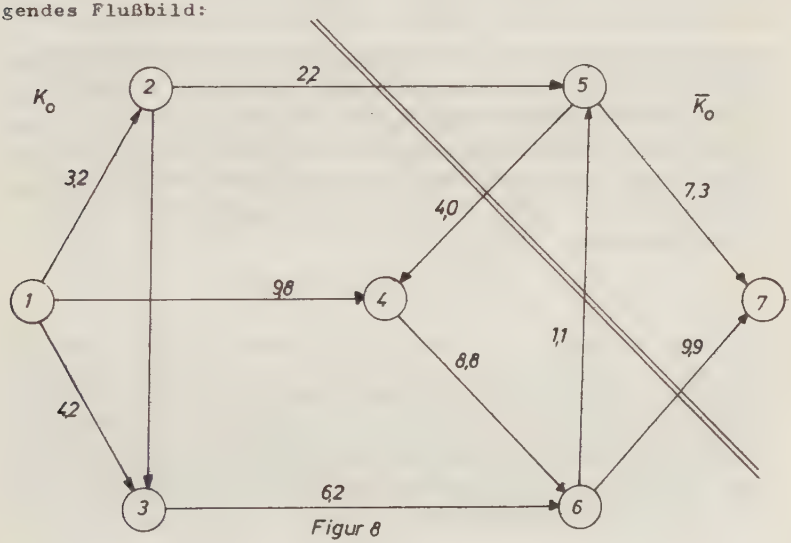
und weiter aus $\curvearrowright$ $g_3 < 0$ $\curvearrowright$ $x_3^0 = 1$

sowie $g_2 < 0$ $\curvearrowright$ $x_2^0 = 1$

Mit $x_1^0 = 1$ und $x_7^0 = 0$ lautet der vollständige Lösungsvektor

$$(1,1,1,1,0,1,0)$$

Der minimale Schnitt ist mit $K_0 = \left\{x_1^0, x_2^0, x_3^0, x_4^0, x_6^0\right\}$, $\overline{K}_0 = \left\{x_5^0, x_7^0\right\}$ festgestellt. Sein Wert ist $C(K_0, \overline{K}_0) = -12$, positiv ist er gleichzeitig der Wert des maximalen Flusses durch das Netzwerk. Er besteht aus den positiven Kapazitäten der geschnittenen Kanten $(x_2,\overline{x}_5)$, $(x_6,\overline{x}_5)$, $(x_6,\overline{x}_7)$. Diese Kapazitäten sind voll ausgelastet, die entgegengerichtete Kante $(x_5,\overline{x}_4)$ bleibt ungenutzt. Im einzelnen ergibt sich folgendes Flußbild:



Figur 8

Nebenstehende Matrix gibt den Flußwert für die einzelnen Teilströme des Netzwerkes an[1].

S_{ij}	1	2	3	4	5	6	7
1	0	2	2	8	0	0	0
2	0	0	0	0	2	0	0
3	0	0	0	0	0	2	0
4	0	0	0	0	0	8	0
5	0	0	0	0	0	0	3
6	0	0	0	0	1	0	9
7	0	0	0	0	0	0	0

Die Bestimmung des maximalen Flusses mit Hilfe der PSEUDO-BOOLEschen Programmierung stellt unserer Ansicht nach gegenüber der Baum- bzw. Kennzeichnungsmethode einen wesentlichen Fortschritt dar, zumal man bei umfangreichen Netzwerken bei der Kennzeichnung sehr schnell die Übersicht verliert, während die Baum-Methode demgegenüber langwierig erscheint[2].

Im Rahmen dieser Arbeit haben wir die PSEUDO-BOOLEsche Programmierung bewußt auf Probleme beschränkt, deren Lösung durch die Minimierung eines komplexen Ausdrucks gewonnen werden konnte, Beschränkungen des Problems sind dabei implizit berücksichtigt worden. Das ist nicht immer möglich und es ergibt sich die Frage, wie Beschränkungen des Lösungsbereichs explizit berücksichtigt werden können, z.B. die Bedingungen

$$f_i(x_1,\ldots,x_n)\ R_i\ 0$$

---

[1] Der Verlauf der Teilströme in K_0 ist nicht immer eindeutig, in unserem Beispiel könnte abweichend auch $(x_1,\bar{x}_2)$ mit drei Einheiten ausgelastet werden, dann müßte $(x_2,\bar{x}_5)$ eine Einheit aufnehmen und $(x_1,\bar{x}_4)$ oder $(x_1,\bar{x}_3)$ eine Einheit weniger. Das sind zusätzliche Wahlmöglichkeiten.

[2] Vgl. die Beschreibung des Verfahrens bei Dantzig, G.B.: Lineare Programmierung und Erweiterungen, a.a.O., S. 452 ff.

bei der Minimierung von

$$f_0(x_1,\ldots,x_n), \text{ wobei } R_i$$

die Relationen $=$, $\leqq$, $\geqq$, $<$, $>$ darstellt. Im Simplex-Algorithmus werden Zielfunktion und Beschränkungen praktisch uno actu gelöst, in der PSEUDO-BOOLEschen Programmierung sind mehrere Schritte erforderlich.

Tatsächlich wird die praktische Handhabung des Algorithmus von IVANESCU durch die zusätzliche Berücksichtigung derartiger Beschränkungen behindert, so daß seine Vorteilhaftigkeit infolge des großen Rechenaufwandes teilweise wieder aufgehoben wird. Im folgenden Kapitel wird ein besonderes Verfahren - der "Additive Algorithmus" von BALAS - dargestellt, welcher uns für entsprechende Problemstellungen geeigneter erscheint. Lediglich der Vollständigkeit halber und wegen der theoretischen Beziehung soll an dieser Stelle der Lösungsweg von IVANESCU angedeutet werden[1].

Zunächst wird jede Beschränkung, was stets möglich ist, in eine Gleichung überführt. Bei der Einführung von Hilfsvariablen ist hier jedoch zu beachten, daß diese wie die Basisvariablen als BOOLEsche Variable definiert werden müssen, entsprechend sind für jede Beschränkung mehrere Hilfsvariable einzuführen[2]. Wenn für

$$f(x_1,\ldots,x_n) \leqq 0 \qquad (1)$$

die Summe der negativen Koeffizienten in der Polynomdarstel-

---

[1] Ausführlich mit Beweisen bei Ivanescu, P.L.: Systems of Pseudo-Boolean Equations and Inequalities, in: Bulletin De L'Academie Polonaise des Sciences, Vol. XII 1964 S.673 ff.

[2] Vgl. zur Binärdarstellung von ganzen Zahlen Abschnitt 31.

lung von f gleich A ist und mit a der kleinste ganzzahlige Wert bezeichnet wird, für den $-A \leqq 2^a$ gilt, kann aus (1) die PSEUDO-BOOLEsche Gleichung

$$f(x_1,\ldots,x_n) + 2^0y_0 + 2^1y_1 + \ldots + 2^ay_a = 0 \quad (2)$$

entwickelt werden. Dabei ist jede Lösung (2) eine Lösung von (1). Für diese Gleichung wird eine spezielle Lösung ermittelt. Dazu stehen verschiedene Möglichkeiten zur Verfügung[1]. Aus der speziellen Lösung können in einem zweiten Schritt alle zulässigen Lösungen entwickelt werden. In einem weiteren Schritt wird die PSEUDO-BOOLEsche Zielfunktion im Bereich der zulässigen Zuordnungen nach der oben dargestellten Methode minimiert.

44. Der "Additive Algorithmus" von BALAS

441. Einführung

Im Abschnitt 41. haben wir für die Lösung PSEUDO-BOOLEscher Programme zwei verschiedene Möglichkeiten angedeutet: Anwendung des Simplex-Algorithmus nach Einführung zusätzlicher, die außerordentliche Ganzzahligkeit betreffender Beschränkungen oder Aufzählung und Prüfung aller möglichen BOOLEschen Lösungsvektoren (sogenannte 'enumeration schemes'). Der hier

---

[1] Vgl. z.B. Löwenheim, L.: Über die Auflösung von Gleichungen im logischen Gebietekalkül, in: Mathematische Annalen 68. Band 1910, S. 169 und
Ivanescu, P.L.: Systems of Pseudo-Boolean Equations and Inequalities, a.a.O., S. 677 f. und
Ivanescu, P.L.: Pseudo-Boolean Programming ..., a.a.O., S. 15.

darzustellende Algorithmus von EGON BALAS[1] stellt als "abbreviated enumeration procedure"[2] den letzten Stand der Entwicklung in der zweiten Richtung dar. BALAS hat für das von Zielfunktion und Beschränkungen gebildete lineare System Kriterien entwickelt, deren Effizienz es erlaubt, bereits nach Durchprüfung einer vergleichsweise sehr kleinen Auswahl der 2^n möglichen 0,1 - Zuordnungen die optimale anzugeben beziehungsweise nachzuweisen, daß eine solche nicht existiert. Neben seiner guten Konvergenz bietet der Additive Algorithmus - so genannt, weil nur additive und subtraktive Operationen erforderlich sind - eine Reihe nützlicher Vorteile, die insbesondere für ökonomische Fragestellungen Bedeutung haben. Vor allem ist dieser Algorithmus im Gegensatz z.B. zu dem Ansatz von FORTET ein allgemeines Verfahren für alle Programme mit BOOLEschen Variablen, ähnlich wie es der Simplex-Algorithmus für reelle Werte ist.

442. Darstellung des Algorithmus[3]

In unserer erläuternden Darstellung gehen wir vom

---

[1] Vgl. Balas, E.: Un algorithme additif pour la resolution des programmes lineaires en variables bivalentes, in: C.R. Acad. Sc. Paris, 258. Bd. 1964, S. 3817 ff.
Balas, E.: Extension de l'algorithme additif a la programmation en nombres entiers et a la programmation non lineaire, in: C.R. Acad. Sc. Paris, 258. Bd. 1964, S. 5136 ff.
Balas, E.: An Additive Algorithm ..., a.a.O., S.517 ff.
Glover, Fred; Zionts, Stanley: A Note on the Additive Algorithm of Balas, in: JORSA Vol. 13 1965 No. 4,S.546 ff.

[2] Balas, E.: An Additive Algorithm ..., a.a.O., S. 518.

[3] Da es sich bei den Veröffentlichungen von Balas um mathematische Spezialliteratur handelt, hier die ökonomische Relevanz im Vordergrund steht, sind zur besseren Einsicht einige formelle Änderungen vorgenommen worden. Besondere mathematische Operationen werden durch Beschreibung ihres realen Hintergrundes erklärt. Beweise, insbesondere über die Konvergenz des Algorithmus können nachgelesen werden bei
Balas, E.: An Additive Algorithm ..., a.a.O., S. 529 ff.

System[1]

$$cx' = \text{Min.!} \qquad (1)$$

$$Ax'+y' = b' \qquad (2)$$

$$x_j = 0 \text{ oder } 1 \in (j \quad N) \qquad (3)$$

$$y \geqq 0 \qquad (4)$$

aus, worin die Zielfunktion z als skalares Produkt der Vektoren $c = (c_1,\ldots,c_n)$ und $x' = (x_1,\ldots,x_n)$ gebildet wird. A ist die Koeffizientenmatrix der Beschränkungsgleichungen und b' ist der m-dimensionale Vektor der geforderten Mindestmengen, y' ist der Vektor der Schlupfvariablen, welche für zulässige Lösungen nicht negativ sein dürfen. Die Lösungsvektoren u = (x,y) haben die Dimension m + n.

Als Ausgangspunkt der Rechnung wird die Nullösung $u^0 = (x^0,y^0) = (0,b)$ fixiert, welche (2) und (3) entspricht und im dualen Bereich zulässig ist[2]. Da das Auffinden der optimalen Lösung, für die (1), (2), (3) und (4) erfüllt sein müssen durch sukzessives Aufnehmen einzelner Variabler in die Lösung geschieht und diese mit dem Wert 1 fixiert werden, beschreibt man den Algorithmus zweckmäßigerweise durch Mengen von Indizes der in die Lösung aufgenommenen Variablen[3].

---

[1] Das bedeutet keine Einschränkung der Allgemeingültigkeit, denn jede Aufgabe mit der Form cx' = z und Ax' $\leqq$ b' für x_j = 0 oder 1 kann als Minimumaufgabe mit kleiner-gleich Bedingungen formuliert werden. Zu den erforderlichen Operationen vgl. Angermann, A.: Entscheidungsmodelle, a.a.O., S. 204 ff.,
Dantzig, G.B.: Lineare Programmierung und Erweiterungen, a.a.O., S. 139 ff.
Balas, E.: An Additive Algorithm, ..., a.a.O., S. 519.

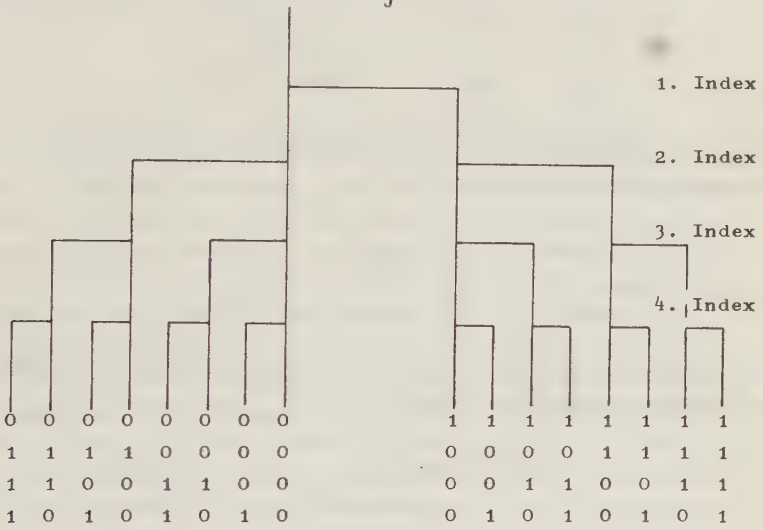
[2] Zum primalen und dualen Problem
vgl. Dantzig, G.B.: Lineare Programmierung und Erweiterungen, a.a.O., S. 144 ff.

[3] Dieses Vorgehen ist typisch für den Additiven Algorithmus, vgl. Balas, E.: An Additive Algorithm ..., a.a.O., S. 519 ff.

Die Menge aller n Variablen x_j wird durch die Indexmenge $N = \{1,\ldots,n\}$ angegeben. Für die Nullösung gilt

$$x_j = 0 \qquad (j \in N)$$

Die Menge aller möglichen Lösungen bildet, geordnet nach den einzelnen Rechenstufen, einen sogenannten Lösungsbaum, wie ihn Figur 9 zeigt, z.B. $x_j = 0$ (j=1,..,4)



Figur 9

Für die Entwicklung des Algorithmus muß das durch (1), (2), (3) und (4) festgelegte System näher beschrieben werden. Häufig tritt das Problem einer zu minimierenden Zielfunktion unter der Beachtung linearer Nebenbedingungen in der Form auf:

$$K = k_1x_1 + k_2x_2 + \ldots + k_nx_n \longrightarrow \text{Min.!}^1$$

[1] $K = (k_1,\ldots,k_n)$ als Stückkostenvektor der n Produkte.

$$A'x_j \geqq b'$$

$$\text{mit } x_j = 0 \text{ oder } 1$$

Explizit im Koeffizientenschema gilt:

$$k_1x_1 + \ldots + k_nx_n = \text{Min.!}$$

$$\begin{aligned} \text{mit } & a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \geqq b_1 \\ & \ldots\ldots\ldots\ldots\ldots\ldots \\ & a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \geqq b_m \end{aligned}$$

Für den Algorithmus werden die Ungleichungen durch Multiplikation mit (-1) umgeformt

$$\begin{aligned} & -a_{11}x_1 - \ldots - a_{1n}x_n \leqq -b_1 \\ & \ldots\ldots\ldots\ldots\ldots\ldots \\ & -a_{m1}x_1 - \ldots - a_{mn}x_n \leqq -b_m \end{aligned}$$

Hierfür setzen wir $Ax \leqq b$ und definieren nach Einführung der Schlupfvariablen y das System

$$Ax + y = b \qquad (2)$$

$$x_j = 0 \text{ oder } 1 \qquad (3)$$

$$y_i \geqq 0 \qquad (4)$$

Für die Nullösung mit $u^0 = (x^0, y^0) = (0,b)$ ist (4) in der Regel nicht erfüllt, da zumindest einige y_i negativ sind, wodurch angegeben ist, daß der dazugehörige Lösungspunkt außerhalb des Beschränkungsraumes liegt. Da jede in die Lösung aufgenommene Variable nur den Wert 1 haben kann und die unberücksichtigten Variablen gleich 0 gesetzt werden, beschreiben wir wie angedeutet die Lösungen auf den einzelnen Stufen nur durch die in ihnen enthaltenen Indizes, so daß z.B. für die Nullösung gilt

$$x_j^0 = 0 \quad (j \in N) \quad \text{und} \quad x_j^0 = 1 \quad (j \in J_0, J_0 = \emptyset)$$

und allgemein für die Lösung nach dem s'ten Rechenschritt

$$x_j^s = \begin{matrix} 1(j \in J_s) \\ 0\left[j \in (N-J_s)\right] \end{matrix}$$

J_s wird dabei als die Teilmenge derjenigen Indizes aus der Menge aller Indizes $N = \{1,\ldots,n\}$ gebildet, die nach dem s'ten Rechenschritt in die Lösung aufgenommen worden sind, d.h.

$$J_s = \left\{ j \mid j \in N;\ x_j^s = 1 \right\}$$

$$\text{und } u^s = u(j_1,\ldots,j_r) = u(J_s) = (x^s, y^s)$$

Das eigentliche Rechenverfahren beginnt mit der Prüfung der Ausgangslösung; es wird untersucht, ob unter den y_i^0 einige sind, die kleiner als 0 sind, ob also noch ein Abstand zum zulässigen Bereich besteht. Diesen Abstand kann man als komplementären Schlupf bezeichnen: $y_i^0 < 0?$ $(i \in M)$. $M = \{1,\ldots,m\}$ ist die Menge der Koeffizientenindizes, durch welche die Zeilen gekennzeichnet werden, die also durch die Anzahl der Beschränkungen fixiert wird. Sofern ein derartiger Schlupf nicht existiert, kann keine bessere zulässige Lösung gefunden werden und der Rechengang ist abzubrechen. Der Lösungsbereich umfaßt in diesem Fall auch den Nullpunkt. In der Regel existiert mindestens ein i_1, für das $y_{i_1}^0 < 0$, meistens mehrere. Die negativen y_i zeigen an, daß man noch außerhalb des zulässigen Bereichs ist.

Im zweiten Rechenschritt müssen nun Variable bestimmt werden, um den Zulässigkeitsbereich möglichst schnell zu erreichen. Jede auf diesem Wege fixierte Lösung muß zumindest dual-zulässig sein. Nicht alle Variablen sind unter diesem Gesichtspunkt für die Aufnahme in die Lösung geeignet, aus N ist eine Teilmenge N_s (s als laufender Index für die einzelnen Rechenschritte) geeigneter Variabler zu ermitteln. N_s beziehungsweise N_0 nach der Nullösung ist eine Restmenge,

welche sich nach der Eliminierung ungeeigneter Variabler ergibt.

Ungeeignet ist zunächst die Menge der schon in einem vorherigen Rechengang für die Lösung ausgewählten Variablen C^s. Sie ist in der Ausgangssituation definitionsgemäß leer ($C^0 = \emptyset$). Ebenfalls ungeeignet ist jede Zuordnung, die im Sinne der Zielfunktion nicht mindestens ebenso gut ist, wie die gerade ermittelte. Es ist also die Indexmenge D^s der Variablen auszuscheiden, welche die vorliegende Lösung verschlechtern würden. Für die Nullösung, die für die Zielfunktion den Wert $z_0 = 0$ fixiert, ist eine Verschlechterung nicht möglich, d.h. $D^0 = \emptyset$. Schließlich ist jede Variable x_j für die Aufnahme in die Lösung ungeeignet, wenn dadurch der oben beschriebene komplementäre Schlupf vergrößert würde, d.h. negative y_i^s noch größer würden ($\hat{=} |y_i^s|$ wird größer). Auf diese Weise nämlich würde man sich wieder vom zulässigen Bereich entfernen. Damit scheidet die Indexmenge E_s aus, für welche die a_{ij} im Koeffizientenschema positiv oder Null sind, es wird also keine Variable in die Lösung aufgenommen, wenn dadurch auch nur ein y_i vergrößert wird. Das Kriterium für die Zugehörigkeit zu E_s ist $y_i^s \Rightarrow a_{ij} \geqq 0$. Als Teilmenge der die Lage des Lösungspunktes verbessernden, d.h. fortan zu betrachtenden Variablen verbleibt damit

$$N_0 = N - (C^0 \vee D_0 \vee E_0)$$

Da C^0 und D_0 leer sind, bestimmt zunächst E_0 allein den Fortgang der Rechnung. Für jede Zeile der Koeffizientenmatrix werden alle negativen a_{ij} aufaddiert, deren j ein Element von N_0 ist, diesen Summen werden die y_i der gleichen Zeile gegenübergestellt[1]:

$$\sum_{j \in N_0} a_{ij} \leqq y_i^0 \quad ; \quad (i \; y_i^0 < 0)$$

[1] Vgl. Balas, E.: An Additive Algorithm ..., a.a.O., S. 526.

$\overline{a_{ij}}$ symbolisiert die negativen Elemente von A. Wenn die Summe der negativen Koeffizienten einer Zeile, das ist die für diese Beschränkung insgesamt vorhandene Verkürzungsmöglichkeit für den Abstand zum Zulässigkeitsbereich (angegeben durch das y_i), kleiner ist als dieser Abstand, dann kann der zulässige Lösungsbereich nicht erreicht werden. Sofern die angegebene Bedingung jedoch erfüllt ist, wird geprüft, ob auch die reine Ungleichung erfüllt ist. Ist das der Fall, werden bestimmte Auswahlkriterien v_j^O wie folgt errechnet:

Für die Indexmenge N_O werden für alle i die Differenzen $(y_i^O - a_{ij})$ gebildet. Sofern sie negativ sind (positive Differenzen bleiben unbeachtet) werden sie in der Menge M_j^{O-} zusammengefaßt:

$$M_j^{O-} = \left\{ i \,\middle|\, y_i^O - a_{ij} < O \right\}$$

Für jedes j aus N_O bildet die Summe aller negativer Differenzen den Wert v_j^O. Ist M_j^{O-} leer, d.h. sind keine negativen Differenzen gegeben, wird $v_j^O = O$ gesetzt:

$$v_j^O = \begin{cases} \sum\limits_{i \in M_j^{O-}} (y_i^O - a_{ij}), & (j \in N_O;\ M_j^{O-} \neq \emptyset) \\ O, & (j \in N_O;\ M_j^{O-} = \emptyset) \end{cases}$$

Die v_j^s können als die Summe der einzelnen Abstandskomponenten der verkürzten Entfernungen zum Zulässigkeitsbereich nach Einführung der ausgewählten Variablen x_j in die Lösung interpretiert werden. Die einzuführende Variable wird dabei in der Weise gewählt, daß v_j^O maximal ist[1]:

$$v_{j_1}^O = \max_{j \; N_O} v_j^O$$

[1] Es ist zu beachten, daß die v_j^s negative Werte sind und demnach ihr kleinster absoluter Wert das Maximum darstellt.

Das ausgewählte $v^0_{j_1}$ ist zu streichen, die Lösung u^1 zu beschreiben und der ^{1}Wert der Zielfunktion zu bestimmen. Die Lösung wird gegeben durch

$$x^1_j = \begin{cases} 1, & j \in J_1 \\ 0, & j \in (N-J_1) \end{cases}$$

J_1 besteht aus der Vereinigung von J_0 (bislang leer) und der neu aufgenommenen Variablen j_1: $J_1 = J_0 \vee \{j_1\}$ Der Wert der Zielfunktion ist gegeben durch $z_1 = z_0 + c_{j_1}$.[1] Die y_i verändern sich bei Berücksichtigung der eingeführten Variablen zu $y^1_i = y^0_i - a_{ij_1}$. Damit ist eine Iteration auf dem Wege zur Optimallösung beendet und die zweite kann entsprechend durchgeführt werden. Dazu wird zunächst der Vektor der y_i untersucht, ob noch ein $y^1_i < 0$ ist. Wenn ja, kann die Menge N_1 der zur Aufnahme in die ^{1}Lösung u^1 in Frage kommenden Variablen bestimmt werden mit

$$N_1 = N - (C^1 \vee D_1 \vee E_1)$$

Die Indexmenge C^1 ist für den zweiten Rechenschritt nicht mehr leer, da für u^1 die Variable x_{j_1} in die Lösung aufgenommen worden ist und nicht noch einmal berücksichtigt werden darf. D_1 als die Menge der Indizes, welche bei Einführung in die Lösung das Ergebnis verschlechtern würden - wenn dieses im zulässigen Bereich schon besser war - , ist wiederum leer, da wir uns dem zulässigen Bereich erst nähern. Es gilt:

$$D_s = \left\{ j \,\middle|\, j \in (N-C^s),\ c_j \geq z^{*(s)} - z_s \right\}$$

Hierin ist $z^{*(s)}$ der minimale (d.h. der bislang beste) Wert der Zielfunktion für alle vergangenen zulässigen Lösungen.

---

[1] Allgemein gilt für den Wert der Zielfunktion
$z_s = cx' = c_1x_1 + c_2x_2 + \ldots + c_sx_s$.

Bezeichnet man mit Z_s die Menge aller Zielfunktionswerte z_p im zulässigen Bereich, so ist

$$Z_s = \left\{ z_p \,\middle|\, p \leqq s,\ u^p \geqq 0 \right\}$$

worin $u^p \geqq 0$ angibt, daß man im zulässigen Bereich ist. Die Grenzen für den Wert $z^{*(s)}$ aller z_p definiert BALAS[1] mit

$$z^{*(s)} = \begin{cases} \infty\,, & \text{für } Z_s = \emptyset \\ \min z_p \in Z_s & \text{für } Z_s \neq \emptyset \end{cases}$$

Solange man sich also noch nicht im zulässigen Lösungsbereich befindet, ist die oben definierte Menge D_s leer, denn für den nicht zulässigen Bereich wird aus $c_j \geqq z^{*(s)} - z_s$ $c_j \leqq -z_s$.

E_s kann einige Indizes enthalten, berücksichtigt werden nur solche, die nicht schon in C^s oder D^s enthalten sind, bzw. den komplementären Schlupf vergrößern würden. Es ist:

$$E_s = \left\{ j \,\middle|\, j \in \left[N - (C^s \vee D_s) \right],\ y_i^s < 0 \Rightarrow a_{ij} \geqq 0 \right\}$$

Nachdem N_1 ermittelt ist, werden für alle darin enthaltenen Indizes die Summen der negativen Koeffizienten einer Zeile mit den entsprechenden y_i^1 verglichen:

$$\sum_{j \in N_1} a_{ij} \leqq y_i^1,\ (i \,|\, y_i^1 < 0)$$

Wenn auch die strenge Ungleichung erfüllt ist, sind wiederum die Werte v_j^1 als Auswahlkriterien für die Variablen zu berechnen:

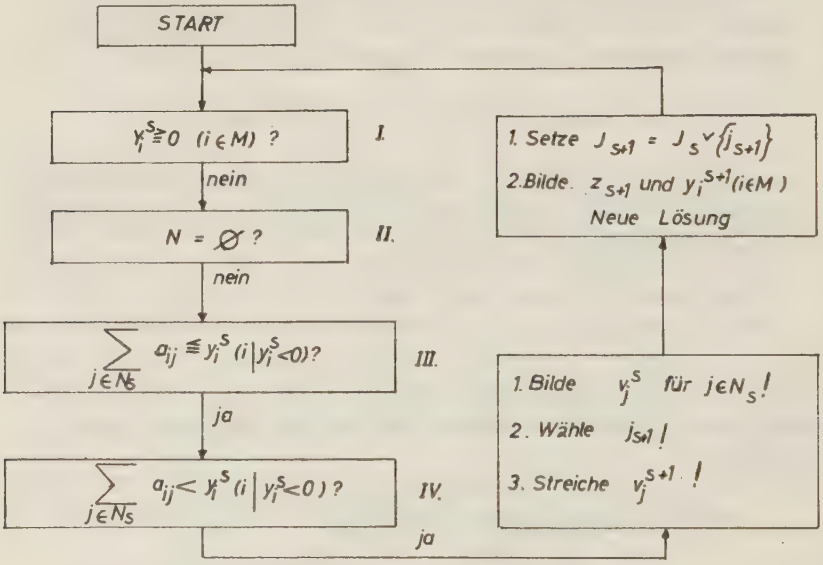
$$v_j^1 = \begin{cases} \displaystyle\sum_{i \in M_j^{1-}} (y_i^1 - a_{ij}) & (j \in N_1;\ M_j^{1-} \neq \emptyset) \\ 0 & (j \in N_1;\ M_j^{1-} = \emptyset) \end{cases}$$

[1] Vgl. Balas, E.: An Additive Algorithm ..., a.a.O., S. 523.

Aus diesen v_j^1 wird wieder das Maximum bestimmt, entsprechend der Anweisung von BALAS gestrichen und der dazugehörige Index in die Lösung aufgenommen: $J_2 = J_1 \vee \{j_2\}$. Der Wert der Zielfunktion wird zu $z_2 = z_1 + c_{j_2} = c_{j_1} + c_{j_2}$. Die korrigierten Werte für y_i ergeben sich als Differenzen

$$y_i^2 = y_i^1 - a_{ij_2} = y_i^0 - \sum_{j \in J_2} a_{ij}$$

Die in dieser Weise dargestellten Iterationen lassen sich eine gewisse Zeit lang fortführen. Durch sie wird jedoch nur ein möglicher Weg durch den Lösungsbaum, d.h. der Rechenvorgang für eine ganz bestimmte Aufgabe gezeigt. Das dazugehörige Flußbild ist in Figur 10 dargestellt.



Figur 10

Um den Algorithmus nunmehr vollständig zu beschreiben, müssen bei der Überprüfung der Kriterien für die vorzunehmenden Rechenschritte die möglichen Alternativen berücksichtigt werden. Tatsächlich stellt nämlich jeder der bisher durchgeführten Prüfschritte (Befragungen) in der Folge der Iterationen eine Weiche für die Alternativen Ja - Nein dar, von denen nur jeweils eine berücksichtigt wurde.

Ist also z.B. unter den Werten y_i^s $(i \in M)$ nach s Rechenschritten keiner mehr kleiner als Null, so ist der Zulässigkeitsbereich erreicht worden[1].

Folgende Maßnahmen sieht der Algorithmus in diesem Fall vor: Es ist die Indexmenge D_k^s zu bilden $(k<s)$. Dazu betrachtet man die vorliegende letzte Lösung u^s und eine der weiter zurückliegenden Lösungen u^k $(k<s)$. Zur Lösung u^k gehören bestimmte Werte v_j^k. Bis zum Erreichen der Lösung u^s sind von diesen v_j^k einige Werte gestrichen worden. Die Menge aller j, für welche die Werte v_j^k der Lösung u^k auf dem Wege zur Lösung u^s gestrichen worden sind, wird mit C_k^s bezeichnet, C_k^k ist naturgemäß leer. Für die Lösung u^k bildet man die Indexmenge $N_k - C_k^s$. Aus dieser Menge werden die j ausgewählt, für welche der Wert der Zielfunktion z_{s+1} die Obergrenze erreichen würde, wenn c_j in die Lösung aufgenommen würde. Man hat

$$J_{s+1} = J_k \vee \{j\} \text{ mit } D_k^s = \left\{ j \,\middle|\, j \in (N_k - C_k^s),\ c_j \geqq z^*(s) - z_k \right\}$$

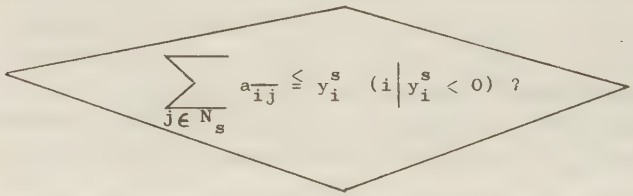
Zur näheren Erläuterung diene eine weitergehende Erklärung der einzelnen Schritte. Nach drei Iterationen (s=3) sei die Form erreicht, daß kein y_i^3 mehr negativ ist, d.h.

$$y_i^3 = y_i^0 - \sum_{j \in J_3} a_{ij} \geqq 0 \text{ für alle } i \in M.$$

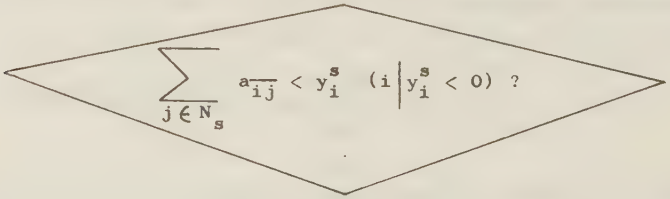
[1] Es ist zu beachten, daß man aus dem dual-zulässigen Bereich in den Zulässigkeitsbereich eingedrungen ist, sich in diesem also zwangsläufig an einem Punkt befinden muß, der nicht mehr sehr weit vom Optimum entfernt sein kann. Sobald man einen Punkt im Zulässigkeitsbereich ermittelt hat, verbleibt als Aufgabe die Annäherung an das Optimum.

Weiterhin sei $z_3 = z^{*(3)}$ die bislang günstigste Lösung. Jetzt werden die Mengen D_k^3 für k = 2,1,0 gebildet. Bestand N_2 z.B. aus $\{j_4, j_5, j_6\}$ und C_2^3 aus $\{j_4\}$, so berechnet man den Wert der Zielfunktion für k = 2 (z_2) und vergleicht für die Indexmenge $N_2 - C_2^3 = \{j_5, j_6\}$ die Komponenten der Zielfunktion mit der Differenz $z^{*(3)} - z_2$. Ist hierbei sowohl $c_{j_5} < z^{*(3)} - z_2$ als auch $c_{j_6} < z^{*(3)} - z_2$, so ist $D_2^3 = \emptyset$. Entsprechend müssen D_1^3 und D_0^3 bestimmt werden. Für alle Indizes, welche in diesen Teilmengen D_k^s enthalten sind, werden die dazugehörigen v_j^k gestrichen. Im Anschluß hieran folgt ein völlig anderer Abschnitt im Rechenverfahren, durch welchen das Ende des Algorithmus festgestellt wird. Zu diesem Abschnitt gehören die Alternativen der nächsten beiden Iterationsschritte.

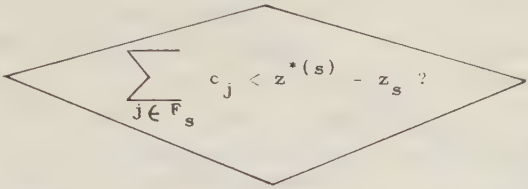
Die Frage der zweiten Weiche lautete: $N_s = \emptyset$? Für "Nein" folgt der dritte Weichenpunkt, für "Ja" schließt sich die Überprüfung an, welche auf die Berechnung der D_k^s folgt und noch zu beschreiben ist. Die Frage des dritten Verzweigungspunktes ist:



Bei negativer Antwort wird man ebenfalls zu der bereits erwähnten Überpfüfung hingeleitet, bei Bejahung folgt der bereits dargestellte vierte Rechenschritt (vgl. Figur 11).
Die Ja-Alternative für die vierte Frage



ist bereits durchgespielt. Lautet die Antwort "Nein"[1], so führt BALAS einen Vergleich der Komponenten der Zielfunktion durch. Es wird geprüft, ob

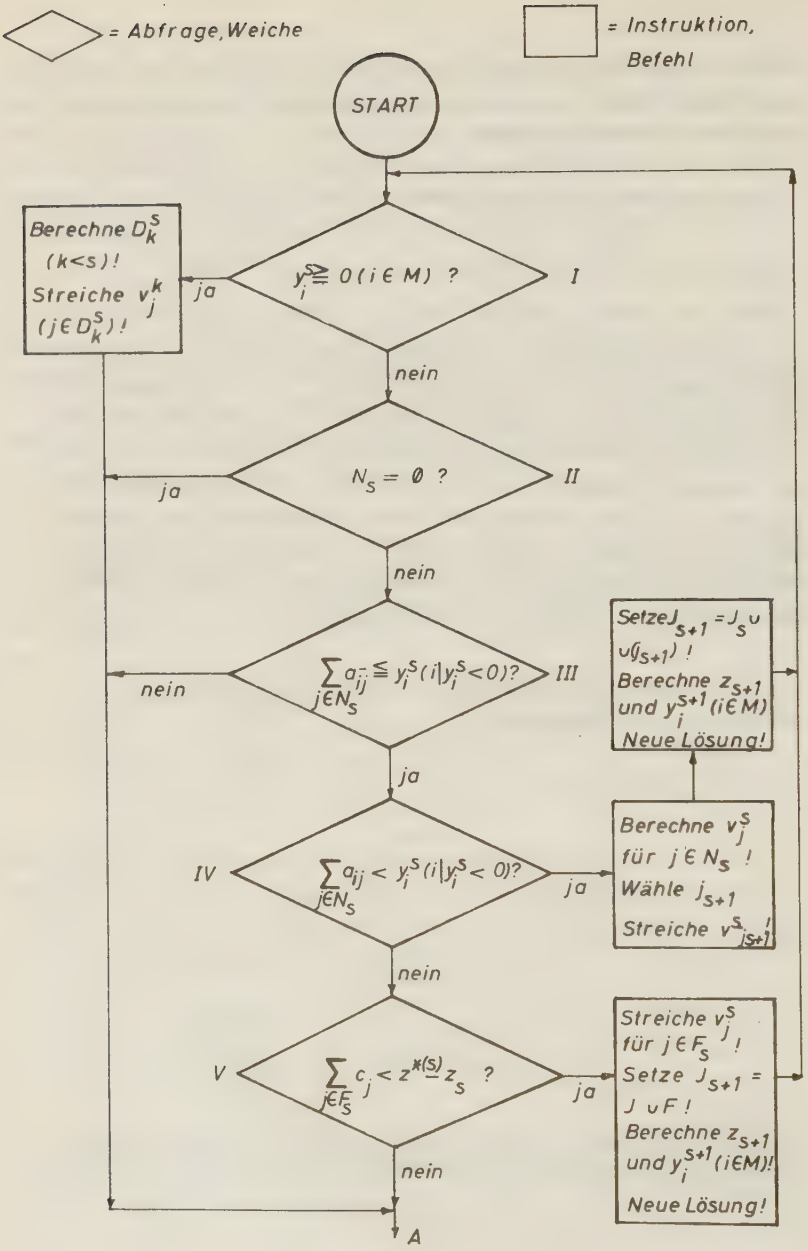


F_s ist die Teilmenge von Indizes aus N_s, für die für mindestens ein i aus N_s der Koeffizient $a_{ij} < 0$ ist. Wird die Ungleichung erfüllt, lautet die Antwort also 'Ja', sind die v_j^s für die Variablen der Indexmenge F_s zu streichen, F_s in die Lösung aufzunehmen: $J_{s+1} = J_s \vee F_s$. Weiter bestimmt man z_{s+1} und y^{s+1} und beginnt den Iterationsvorgang von neuem. Lautet die Antwort dagegen "Nein", so sind die v_j^s für alle $j \in N_s$ zu streichen und die erwähnte noch zu beschreibende Überprüfung der Lösung ist vorzunehmen.

Das Flußdiagramm unter Berücksichtigung der Alternativen ist in Figur 11 dargestellt.

---

[1] Diese Verneinung bedeutet, daß die Summe aller negativen Koeffizienten gerade gleich dem Abstand zum Zulässigkeitsbereich ist.


= Abfrage, Weiche
= Instruktion, Befehl
START
$y_i^s \geqq 0 (i \in M)$?
I
ja
Berechne D_k^s $(k<s)$! Streiche v_j^k $(j \in D_k^s)$!
nein
$N_s = \emptyset$?
II
ja
nein
$\sum_{j \in N_s} a_{ij} \leqq y_i^s (i | y_i^s < 0)$?
III
nein
ja
Setze $J_{s+1} = J_s \cup \{j_{s+1}\}$! Berechne z_{s+1} und y_i^{s+1} $(i \in M)$ Neue Lösung!
$\sum_{j \in N_s} a_{ij} < y_i^s (i | y_i^s < 0)$?
IV
ja
Berechne v_j^s für $j \in N_s$! Wähle j_{s+1} Streiche $v_{j_{s+1}}^s$!
nein
$\sum_{j \in F_s} c_j < z^{*(s)} - z_s$?
V
ja
Streiche v_j^s für $j \in F_s$! Setze $J_{s+1} = J \cup F$! Berechne z_{s+1} und y_i^{s+1} $(i \in M)$! Neue Lösung!
nein
A

Figur 11

Damit ist geklärt, wie der erste Zyklus des Algorithmus abgeschlossen wird. Der Stand der Rechnung am Punkte A kann durch folgende Einzelheiten beschrieben werden:

1. Es sind keine y_i^s mehr negativ, oder

2. N_s ist leer oder,

3. die Summe der negativen $\overline{a_{ij}}$ einer Zeile kann das negative y^s dieser Zeile nicht erreichen oder,

4. die Summe der negativen $\overline{a_{ij}}$ einer Zeile ist zwar gleich dem y^s, doch die Zielfunktion wird nicht mehr verbessert.

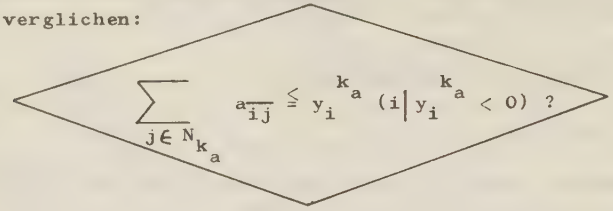
Aufgabe des zweiten Zyklus ist es jetzt, die Optimalität der ermittelten Lösung zu untersuchen. Die nächste Frage dazu lautet: Verbleiben nach s Iterationen für die Lösung u^k $(k \mid J_k \subset J_s)$ noch j, welche die Lösung verbessern können? Die Indexmenge der verbessernden Vektoren soll durch N_k^s beschrieben werden mit k als laufendem Index, so daß für

$$N_k^s = N_k - (C_k^s \vee D_k^s)$$

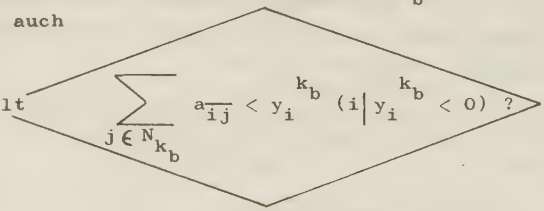
gilt VI.: $N_k^s = \emptyset$ $(k \mid J_k \subset J_s)$?

Ist man z.B. nach drei Rechenschritten bei VI. angelangt, wird N_k^3 für k = 2, d.h. $N_2^3 = N_2 - (C_2^3 \vee D_2^3)$ gebildet. Ist $N_2^3 = \emptyset$, wird $N_1^3 = N_1 - (C_1^3 \vee D_1^3)$ geprüft und falls leer auch $N_0^3 = N_0 - (C_0^3 \vee D_0^3)$ geprüft. Sind alle N_k^s leer, so ist die Lösung u^s die optimale und der Algorithmus ist beendet. Falls Z_s für diesen Punkt ebenfalls leer ist, hat das Problem keine zulässige Lösung. Findet sich jedoch unter den N_k^s ein k_a, für das die Menge nicht leer ist, so werden im nächsten Schritt

die negativen $y_i^{k_a}$ mit den negativen Koeffizienten der dazugehörigen Zeilen und der Spalten[1], deren Indizes zu N_k^s gehören, verglichen:

VII. 

Wird diese Frage verneint, so soll man die $v_j^{k_a}$ für alle j, die zu $N_{k_a}^s$ gehören, streichen und prüfen, ob ggf. noch ein weiteres k existiert, welches kleiner als k_a ist. Ist dieses nicht der Fall, so gibt es auch keine verbessernde Lösung mehr und der Algorithmus ist beendet. Sind jedoch noch kleinere k vorhanden, ist bei VI. erneut einzusetzen. Wird auch VII. bejaht, d.h. ist VII. für ein bestimmtes k_b erfüllt, so wird geprüft ob auch

VIII. gilt 

Wird diese Bedingung erfüllt, existieren andere Lösungen und die entsprechenden Auswahlkriterien $v_j^{k_b}$ sind zu bilden, ein j_{s+1} zu wählen und das dazugehörige $v_{j_{s+1}}^{k_b}$ zu streichen. Nach Bildung der neuen Lösung $J_{s+1} = J_s \vee \{j_{s+1}\}$ berechnet man den Wert der Zielfunktion z_{s+1} und den Schlupf $y_i^{s+1} (i \in M)$, um anschließend erneut bei I. zu beginnen.

---

[1] Dies ist ein gutes Kennzeichen dafür, daß mit zunehmender Annäherung an die optimale Lösung immer empfindlicher werdende Kriterien benutzt werden, damit man nicht auf ungünstigere Zuordnungen zurückkommt.

Wenn die Ungleichung nicht erfüllt ist, besteht Gleichheit zwischen wenigstens einer Summe $\sum_{j \in N_{k_b}} a_{ij}$ und $y_i^{k_b}$. Das weitere Vorgehen ist abhängig von den Komponenten der Zielfunktion, dazu bildet man

IX. $\sum_{j \in F_{k_b}^s} c_j < z^*(s) - z_{k_b}$?

Hierin ist $F_{k_b}^s$ die Teilmenge von $N_{k_b}^s$, für welche die $a_{ij} < 0$ für mindestens ein $i \in M_{k_b}^s$ sind. Wird diese Ungleichung von einer der Komponenten der Zielfunktion mit der Indexmenge $F_{k_b}^s$ erfüllt, so nimmt man sie in die Lösung auf und erhält:

$$J_{s+1} = J_s \vee F_{k_b}^s$$

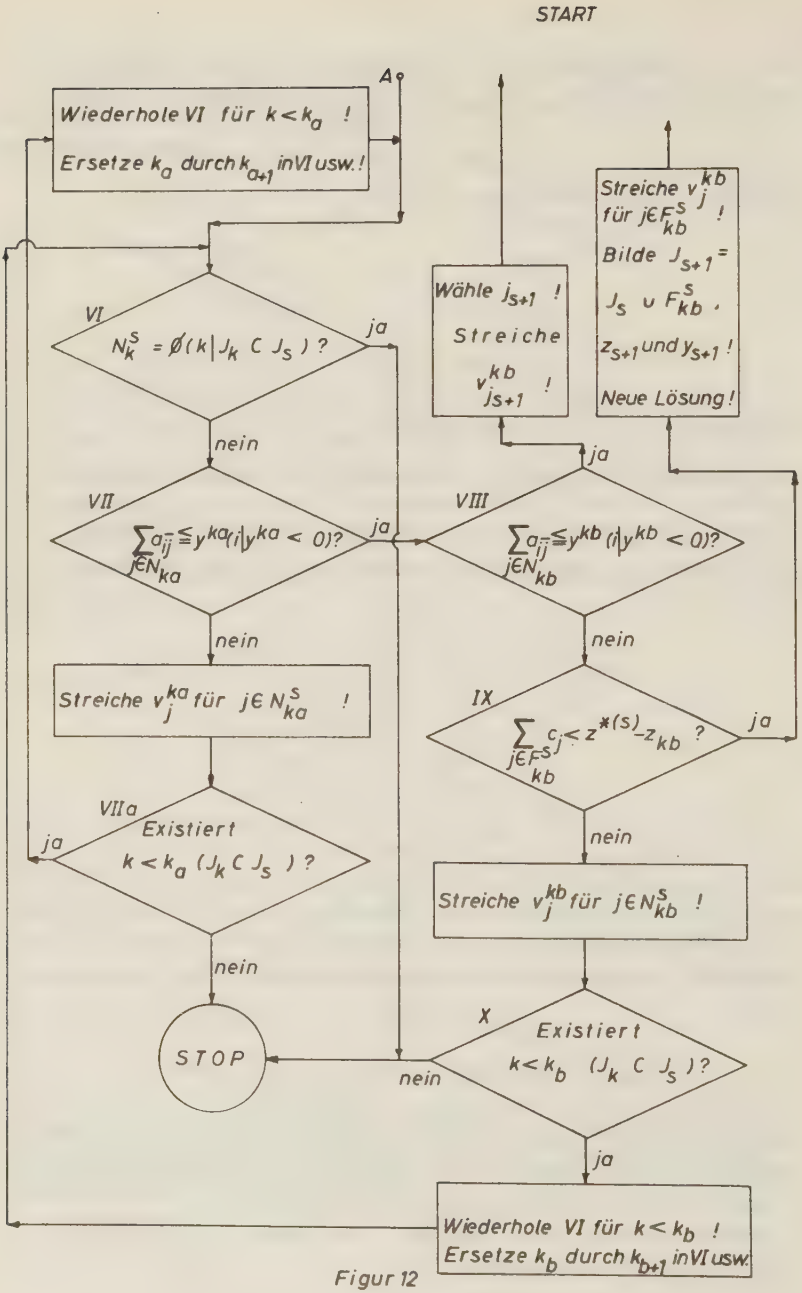
$$z_{s+1} = z_{k_b} + \sum_{j \in F_{k_b}^s} c_j$$

$$y_i^{s+1} = y_i^{k_b} - \sum_{j \in F_{k_b}^s} a_{ij}$$

Wird die Ungleichung IX. nicht erfüllt, so ist dies das Zeichen dafür, daß im untersuchten Zusammenhang keine bessere Lösung gefunden werden kann. Es verbleibt zu prüfen, ob andere $k < k_b$ existieren, um sie entsprechend zu testen. Bleibt auch diese Frage

X. Existieren $k < k_b (J_k \quad J_s)$?

verneint, ist der Algorithmus endgültig abgeschlossen. Wird die Frage bejaht, muß mit der Überprüfung des nächst kleineren k erneut bei VI. begonnen werden. In Figur 12 ist dieser zweite Zyklus, welcher sich an Punkt A in Figur 11 anschließt, übersichtlich wiedergegeben.


START
A
Wiederhole VI für $k < k_a$!
Ersetze k_a durch k_{a+1} in VI usw.!
VI
$N_k^S = \emptyset (k \mid J_k \subset J_S)$?
ja
nein
VII
$\sum_{j \in N_{ka}} a_{ij} \leqq y^{ka} (i \mid y^{ka} < 0)$?
ja
nein
Streiche v_j^{ka} für $j \in N_{ka}^S$!
VIIa
Existiert $k < k_a$ $(J_k \subset J_S)$?
ja
nein
STOP
Wähle j_{s+1} !
Streiche $v_{j_{s+1}}^{kb}$!
VIII
$\sum_{j \in N_{kb}} a_{ij} \leqq y^{kb} (i \mid y^{kb} < 0)$?
ja
nein
IX
$\sum_{j \in F_{kb}^S} c_j < z^{*(s)} - z_{kb}$?
ja
nein
Streiche v_j^{kb} für $j \in N_{kb}^S$!
X
Existiert $k < k_b$ $(J_k \subset J_S)$?
nein
ja
Streiche v_j^{kb} für $j \in F_{kb}^S$!
Bilde $J_{s+1} = J_s \cup F_{kb}^S$,
z_{s+1} und y_{s+1} !
Neue Lösung!
Wiederhole VI für $k < k_b$!
Ersetze k_b durch k_{b+1} in VI usw.

Figur 12

Zusammenfassend sollen die besonderen Kennzeichen des Additiven Algorithmus, die gleichzeitig seine Vorteile ausmachen, aufgezählt werden:

1. Rechentechnisch stellt sich der Additive Algorithmus als eine Abfolge von Ja - Nein-Entscheidungen dar.

2. Als einzige mathematische Operationen sind Additionen und Subtraktionen erforderlich.

3. Verschiedene Tableaus wie bei den herkömmlichen Verfahren der linearen Programmierung gibt es nicht; das Koeffizientenschema bleibt unverändert.

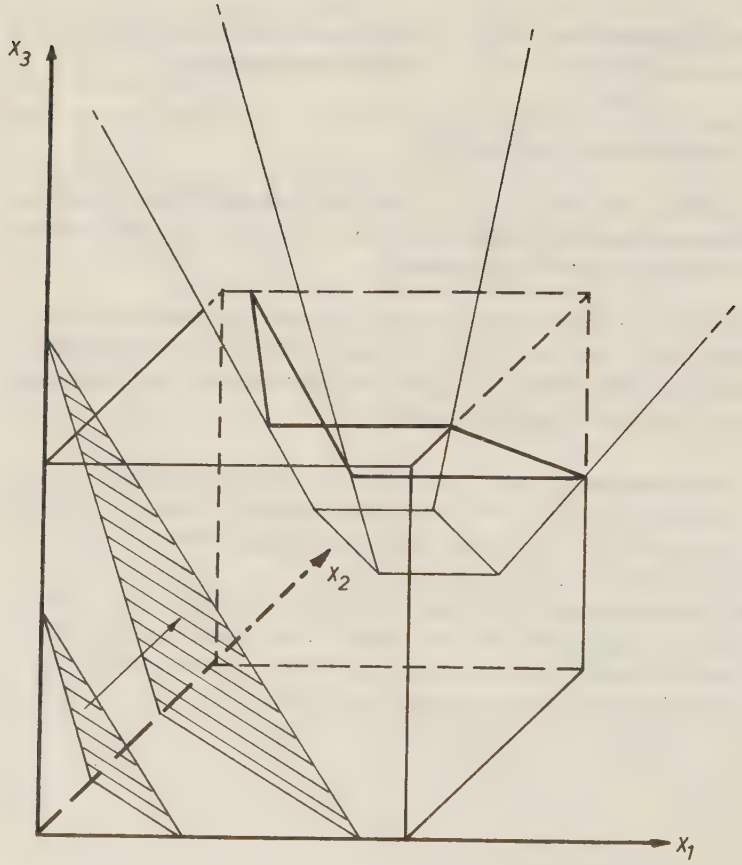
4. Durch besonders effiziente Kriterien wird laufend kontrolliert, ob man sich im primal- oder dual-zulässigen Bereich befindet, bzw. kann die Optimalität der vorliegenden Lösungen überprüft werden.

Die Wirksamkeit der Kriterien bestimmt die Konvergenz des Algorithmus. Diese hat sich auch bei großen Problemen als sehr gut erwiesen, wie die praktische Erfahrung zeigt[1].

Bevor ein praktisches Problem mit Hilfe des Additiven Algorithmus gelöst wird, soll an einer,- der Darstellbarkeit wegen sehr vereinfachten Minimum-Aufgabe - das Vorgehen graphisch erklärt werden. Vgl. hierzu Figur 13.

---

[1] Z.B. waren zur Bestimmung der optimalen Lösung bei Problemen mit 10 bzw. 12 Variablen und 7 bzw. 6 Beschränkungen von den insgesamt $2^{10} = 1024$ bzw. $2^{12} = 4096$ nur 5 bzw. 39 Zuordnungen zu prüfen. Vgl. hierzu
Balas, E.: An Additive Algorithm ..., a.a.O., S. 533 ff.



Figur 13

Der von oben rechts in den Würfel eintauchende rechteckige Pyramidenstumpf stellt die Menge aller zulässigen Lösungspunkte dar. Er wird durch die Beschränkungen gebildet. Die Eckpunkte des Würfels sind die möglichen - in diesem Fall $2^3 = 8$ - Lösungspunkte. Die eingezeichnete Hyperebene veranschaulicht die Zielfunktion. Ohne Berücksichtigung von Ganzzahligkeitsbedingungen ist der optimale Lösungspunkt durch den Berührungspunkt der Hyperebene mit dem Pyramidenstumpf fixiert. Führt man BOOLEsche Ganzzahligkeitsbedingungen ein, verbleibt nur noch ein realisierbarer Lösungspunkt ($x_1=x_2=x_3=1$). Insofern ist das Beispiel trivial. Die Auswahlkriterien des Additiven Algorithmus für die Aufnahme der Variablen in die Lösung gehen - wie wir dargestellt haben - auf die Abstände zum Zulässigkeitsbereich (komplementärer Schlupf) zurück. Das führt dazu, daß von allen möglichen Wegen zu einem zulässigen Lösungspunkt der günstigste ausgewählt wird. Verschiebt man für unser Beispiel den Pyramidenstumpf noch weiter zum Ursprung hin, so daß auch die hintere obere Ecke mit den Koordinaten $x_1 = 0$, $x_2 = x_3 = 1$ zulässig wird, bleibt x_1 von vornherein außer Betracht.

443. Praktische Erprobung des "Additiven Algorithmus"

Nicht nur für Entscheidungsprobleme mit unteilbaren Produktionsfaktoren lassen sich zahlreiche Anwendungsmöglichkeiten für PSEUDO-BOOLEsche Modelle angeben, sondern auch für die Seite der Leistungsverwertung. Mit unteilbaren Produkteinheiten muß man z.B. in der Bauindustrie einschließlich Schiff-, Brücken- und Großmaschinenbau rechnen. Gewöhnlich stehen hierbei vor Aufnahme der Produktion, die in diesen Fällen fast ausschließlich Auftragsproduktion ist, mehrere Projekte zur Auswahl. Die gleichzeitige Ausführung aller Aufträge ist infolge betrieblicher Beschränkungen nur selten möglich. Diese Beschränkungen können in Form kapazitativ bestimmter Höchst- oder Mindestwerte gegeben sein.

Häufig gilt es, unter Beachtung der als Nebenbedingungen formulierten Beschränkungen das gewinnmaximale Produktionsprogramm zu ermitteln.

Ein typisches Beispiel aus der Bauindustrie soll dazu dienen, den Additiven Algorithmus praktisch zu erproben.

Gegeben sei die Zielfunktion

$$8x'_1+6,2x'_2-4,2x'_3+5,8x'_4+6,4x'_5-3,1x'_6+5x'_7+4,5x'_8 = \text{Max.!}$$

mit den Dimensionen $\left[\frac{DM}{Stck.} \cdot Stck. = DM\right]$

Aus den Unterlagen der Arbeitsvorbereitung stammen die Angaben über Faktorverbrauch (z.B. Fertigungszeiteinheiten) pro Erzeugniseinheit und die Mindest- oder Höchstverbrauchsmengen für die einzelnen Faktorarten:

$$2x'_1+7x'_2+3x'_3-7x'_5+2x'_6+5x'_8 \geqq 8$$

$$5,5x'_2+6x'_3-4x'_4+5,2x'_5-3,8x'_7 \geqq 6,5$$

$$-5x'_1+4,5x'_2+9x'_4+4x'_5+5x'_6-5x'_8 \leqq 3,5$$

$$3x'_1-5,2x'_2+3x'_3+6,4x'_4-3,5x'_6+6x'_7 \leqq 1$$

Negative a_{ij}-Werte im Koeffizientenschema haben ihre Erklärung etwa darin, daß bei der Übernahme des betreffenden Auftrages (Projektes) eine Arbeitsgemeinschaft mit einer anderen Unternehmung gebildet werden muß und von dieser in der angegebenen Höhe ($a_{\overline{ij}}$) zusätzliche Leistungen (z.B. Erdarbeiten) übernommen werden müssen. Wenn die gesamte Beanspruchung des betrieblichen Leistungsbereichs für den betreffenden Produktionsfaktor (Erdarbeiten) bei einem bestimmten Produktionsprogramm, welches das gemeinsame Projekt einschließt, ermittelt werden soll, müssen diese zusätzlichen Faktoreinsatzmengen subtrahiert werden. Daß auch verlustbringende Projekte aus den un-

terschiedlichsten Gründen in Erwägung gezogen werden können, bedarf keiner Erklärung.

Als Ausgangspunkt für den Additiven Algorithmus war ein Programm mit einer zu minimierenden Zielfunktion unter kleiner-gleich Bedingungen angegeben worden. Aus diesem Grund wird das ursprüngliche System wie folgt umgeformt[1]:

(i) Zielfunktion und größer-gleich Bedingungen werden mit (-1) multipliziert.

(ii) Für die Variablen im ursprünglichen Programm wird gesetzt bei

$$\text{Minimierung} \quad x_j' = \begin{cases} 1 - x_j, & \text{wenn } c_j < 0, \text{ bzw. bei} \\ x_j & \text{sonst,} \end{cases}$$

$$\text{Maximierung} \quad x_j' = \begin{cases} 1 - x_j, & \text{wenn } c_j > 0, \\ x_j & \text{sonst.} \end{cases}$$

Nach Ausführung dieser Vorschriften erhält man:

$$-8(1-x_1)-6{,}2(1-x_2)+4{,}2x_3-5{,}8(1-x_4)-6{,}4(1-x_5)+3{,}1x_6-5(1-x_7)-4{,}5(1-x_8)=\text{Min.!}$$

$$-2(1-x_1)-7\ (1-x_2)-3x_3 \qquad +7\ (1-x_5)-2x_6 \qquad -5\ (1-x_8) \leqq -8$$

$$-5{,}5(1-x_2)-6x_3 \quad +4\ (1-x_4)-5{,}2(1-x_5) \quad +3{,}8(1-x_7) \qquad \leqq -6{,}5$$

$$-5(1-x_1)+4{,}5(1-x_2) \qquad +9\ (1-x_4)+4\ (1-x_5)+5x_6 \qquad -5\ (1-x_8) \leqq 3{,}5$$

$$3(1-x_1)-5{,}2(1-x_2)+3x_3 \quad +6\ (1-x_4) \qquad -3{,}5x_6+6(1-x_7) \qquad \leqq 1$$

[1] Vgl. Balas, E.: An Additive Algorithm ..., a.a.O., S. 519.

und nach Einführung von Schlupfvariablen y_i

$$8x_1+6,2x_2+4,2x_3+5,8x_4+6,4x_5+3,1x_6+5x_7+4,5x_8 \quad =\text{Min.}!^1$$

$$2x_1+7x_2 \quad -3x_3 \quad -7x_5 \quad -2x_6 \quad +5\,x_8 \quad +y_1 \quad \leqq -1$$

$$5,5x_2-6x_3 \quad -4x_4 \quad +5,2x_5 \quad -3,8x_7 \quad +y_2 \quad \leqq -3,6$$

$$5x_1-4,5x_2 \quad -9x_4 \quad -4x_5 \quad +5x_6 \quad -5x_8 \quad +y_3 \quad \leqq -4$$

$$-3x_1+5,2x_2+3x_3 \quad -6x_4 \quad -3,5x_6-6x_7 \quad +y_4 \leqq -8,8$$

Der Algorithmus beginnt für die Nullösung mit der Prüfung der y_i^0, d.h. mit der Frage, ob man sich im zulässigen Bereich befindet: $y_1^0 = -1$, $y_2^0 = -3,6$, $y_3^0 = -4$, $y_4^0 = -8,8$. Die negativen Werte für die Schlupfvariablen zeigen an, daß man sich noch außerhalb des zulässigen Bereichs befindet. Dieser soll möglichst schnell erreicht werden. Das geschieht durch die sukzessive Aufnahme geeigneter Variabler in die Lösung.

Ungeeignet für die Aufnahme sind alle bereits einmal ausgewählten Variablen, bislang keine, d.h. $C^0 = \emptyset$. Ebenfalls leer ist D_0, da es keine Variable gibt, durch deren Aufnahme die Nullösung verschlechtert werden könnte. E_0 dagegen - als Menge der Variablen, durch deren Aufnahme sich der komplementäre Schlupf noch weiter vergrößern würde - ist nicht leer, denn da für $j = 8$ alle $a_{i8} \geqq 0$ sind, würde bei Ausfall dieser Variablen der Abstand zum Zulässigkeitsbereich noch größer. Man hat $E_0 = \{8\}$ und $N_0 = \{1,2,3,4,5,6,7,8\} - \{8\} = \{1,2,3,4,5,6,7\}$ Für N_0 werden die negativen a_{ij} der einzelnen Zeilen, für die $y_i^0 < 0$ ist, aufaddiert und mit den Schlupfvariablen verglichen.

---

[1] Konstante entfallen, da sie ohne Einfluß auf die Minimalwert-Bestimmung sind.

$$\sum_{j \in N_0} a_{1j} = -3 - 7 - 2 = 12 < y_1^0 = -1$$

$$\sum_{j \in N_0} a_{2j} = -6 - 4 - 3{,}8 = -13{,}8 < y_2^0 = -3{,}6$$

$$\sum_{j \in N_0} a_{3j} = -4{,}5 - 9 - 4 = -17{,}5 < y_3^0 = -4$$

$$\sum_{j \in N_0} a_{4j} = -3 - 6 - 3{,}5 - 6 = -18{,}5 < y_4^0 = -8{,}8$$

a_4

Da alle vier Ungleichungen erfüllt sind, kommt man zur Berechnung der Auswahlkriterien v_j^0 (vgl. Figur 11). Es ist:

$$v_1^0 = \sum_{i \in M_1^{0-}} (y_i^0 - a_{i1}) = (-1-2)+(-3{,}6-0)+(-4-5)+(-8{,}8+3) = -21{,}4$$

$$v_2^0 = \sum_{i \in M_2^{0-}} (y_i^0 - a_{i2}) = (-1-7)+(-3{,}6-5{,}5)+(-8{,}8-5{,}2) = -31{,}1$$

$$v_3^0 = \sum_{i \in M_3^{0-}} (y_i^0 - a_{i3}) = (-4-0)+(-8{,}8-3) = -15{,}8$$

$$v_4^0 = \sum_{i \in M_4^{0-}} (y_i^0 - a_{i4}) = (-1-0)+(-8{,}8+6) = -3{,}8$$

$$v_5^0 = \sum_{i \in M_5^{0-}} (y_i^0 - a_{i5}) = (-3{,}6-5{,}2)+(-8{,}8-0) = -17{,}6$$

$$v_6^0 = \sum_{i \in M_6^{0-}} (y_i^0 - a_{i6}) = (-3{,}6-0)+(-4-5)+(-8{,}8+3{,}5) = -17{,}9$$

$$v_7^0 = \sum_{i \in M_7^{0-}} (y_i^0 - a_{i7}) = (-1-0)+(-4-0)+(-8{,}8+6) = -7{,}8$$

Die Anweisung für die Auswahl der ersten Variablen j_1 lautet: Wähle $v_{j_1}^0$ so, daß v_j^0 der maximale Wert aller Kriterien ist.

Damit ist $j_1 = 4$ und die Lösungsmenge $J_1 = J_0 \vee \{4\} = \{4\}$ fixiert. Als Wert für die Zeilfunktion ergibt sich $z_1 = z_0 + c_{j_1} = 5{,}8$, die korrigierten Schlupfvariablen haben die Werte:

$$y_1^1 = y_1^0 - a_{14} = -1 - 0 = -1$$

$$y_2^1 = y_2^0 - a_{24} = -3{,}6 + 4 = 0{,}4$$

$$y_3^1 = y_3^0 - a_{34} = -4 + 9 = 5$$

$$y_4^1 = y_4^0 - a_{44} = -8{,}8 + 6 = -2{,}8$$

Die zweite Iteration beginnt mit der Bestimmung von N_1 als $N_1 = N - (C^1 \vee D_1 \vee E_1)$. Da $C^1 = \{4\}$, $D_1 = \emptyset$, weil der zulässige Bereich noch nicht erreicht ist, und $E_1 = \{2,8\}$, ist $N_1 = \{1, 3, 5, 6, 7\}$. Für N_1 werden die negativen a_{ij} mit den y_i^1 verglichen, aber nur noch für i = 1,4. Es gilt:

$$\sum_{j \in N_1} a_{\overline{1j}} = -3 - 7 - 2 = -12 < y_1^1 = -1$$

$$\sum_{j \in N_1} a_{\overline{4j}} = -3 - 3{,}5 - 6 = -12{,}5 < y_4^1 = -2{,}8$$

Wieder sind alle Ungleichungen erfüllt und man kommt zur Berechnung der Auswahlkriterien v_j^1:

$$v_1^1 = \sum_{i \in M_1^{1-}} (y_i^1 - a_{i1}) = (-1-2) = -3$$

$$v_3^1 = \sum_{i \in M_3^{1-}} (y_i^1 - a_{i3}) = (-2{,}8 - 3) = -5{,}8$$

$$v_5^1 = \sum_{i \in M_5^{1-}} (y_i^1 - a_{i5}) = (-2{,}8 - 0) = -2{,}8$$

$$v_6^1 = \sum_{i \in M_6^{1-}} (y_i^1 - a_{i6}) = 0, \text{ da } M_6^{1-} = \emptyset$$

$$v_7^1 = \sum_{i \in M_7^{1-}} (y_i^1 - a_{i7}) = (-1 - 0) = -1$$

Als maximaler v_j^1-Wert wird v_6^1 ausgewählt und man hat $j_2 = 6$ sowie $J_2 = J_1 \vee \{j_2\} = \{4,6\}$ und $z_2 = z_1 + c_{j_2} = 8,9$. Für die Schlupfvariablen erhält man die Werte

$$y_1^2 = y_1^1 - a_{16} = -1 + 2 = 1$$

$$y_2^2 = y_2^1 - a_{26} = 0,4 - 0 = 0,4$$

$$y_3^2 = y_3^1 - a_{36} = 5 - 5 = 0$$

$$y_4^2 = y_4^1 - a_{46} = -2,8 + 3,5 = 0,7$$

Nach der zweiten Iteration sind alle Schlupfvariablen $y_i^2 \geq 0$, $N_2 = \emptyset$, wodurch angezeigt wird, daß der zulässige Bereich erreicht ist. Damit ist der erste Zyklus abgeschlossen.

In der zweiten Phase des Algorithmus wird die vorliegende Lösung u^s $(s = 2)$ mit $z^2 = 8,9$ $(\hat{=} z^{*(2)}$, da es der bislang günstigste Zielfunktionswert ist) auf ihre Optimalität untersucht. Zu diesem Zweck bildet man für die zurückliegenden Lösungen u^k $(k = 1,0)$ die Indexmengen $N_k^s = N_k - (C_k^s \vee D_k^s)$:

1. $k = 1$; $N_1 = \{1, 3, 5, 6, 7\}$; $C_1^2 = \{6\}$. Für D_1^2 als Teilmenge aller Indizes, welche in der Restmenge $N_1 - C_1^2$ enthalten sind und für deren Zielfunktionswerte $c_j \geq z^{*(2)} - z_1$ gilt:

$$N_1 - C_1^2 = \{1, 3, 5, 7\}$$

$$\left.\begin{array}{l} c_1 = 8 > z^{*(2)} - z_1 = 3,1 \\ c_3 = 4,2 > 3,1 \\ c_5 = 6,4 > 3,1 \\ c_7 = 5 \quad > 3,1 \end{array}\right\} \quad D_1^2 = \{1,3,5,7\}$$

Damit ist $N_1^2 = \emptyset$, d.h. für k = 1 ist kein Ansatzpunkt für eine andere Lösung gegeben.

2. k = 0; $N_0 = \{1,2,3,4,5,6,7\}$; $C_0^2 = \{4\}$. Für D_0^2 ist zu prüfen, ob $c_j \geq z^{*(2)} - z_0 = 8,9$ ist für j = 1,2,3,5,6,7. Da alle c_j-Werte die gestellte Bedingung für D_0^2 nicht erfüllen, ist $D_0^2 = \emptyset$ und man hat $N_0^2 = \{1,3,5,6,7\}$. Damit ist angezeigt, daß hier eine weitergehende Überprüfung durchgeführt werden muß. Dafür gilt:

$\sum_{j \in N_0^2} a_{ij} \leq y_i^0$ und man hat $-12 < -1$
$-9,8 < -3,6$
$-8,5 < -4$
$-12,5 < -8,8$

alle $y_i^0 > \sum_{j \in N_0^2} a_{ij}$

Aus den v_j^0 $(j \in N_0^2)$ wird der maximale Wert (v_7^1) ausgewählt und gestrichen. Als neue Lösung ergibt sich $J_3 = \{7\}$ mit $z_3 = 5$ und

$$y_1^3 = y_1^0 - a_{17} = -1 - 0 = -1$$
$$y_2^3 = y_2^0 - a_{27} = -3,6 + 3,8 = 0,2$$
$$y_3^3 = y_3^0 - a_{37} = -4 - 0 = -4$$
$$y_4^3 = y_4^0 - a_{47} = -8,8 + 6 = -2,8$$

Daran schließt sich die nächste Iteration wie folgt an:
$y_i^3 < 0$ (i M) für i = 1, 3, 4, $C^3 = \{4, 7,\}$
$D_3 = \{j \mid j \in (N - C^3),\ c_j \geq z^{*(2)} - z_3\} = \{1, 2, 3, 5, 8\}$

$E_3 = \{j \mid j \in [N - (C^3 \vee D_3)],\ y_1^3 < 0 \Rightarrow a_{ij} \geq 0\}$ ist demnach als leer festgestellt und als Indexmenge möglicherweise verbessernder Zuordnungen nach der dritten Iteration verbleibt: $N_3 = N - (C^3 \vee D_3 \vee E_3) = \{6\}$. Für den anschließenden Vergleich der negativen a_{ij}-Werte mit den nach der dritten Iteration noch negativen Schlupfvariablen y_i^3 ergibt sich:

$$a_{16} = -2 \quad < y_1^3 = -1$$

$$a_{36} = 0 \quad < y_3^3 = -4$$

$$a_{46} = -3{,}5 < y_4^3 = -2{,}8$$

Da die geforderte kleiner-gleich Bedingung für y_3^3 nicht erfüllt ist, kommen wir zur Prüfung der Zuordnungen aus N_0^3: $N_0^3 = N_0 - (C_0^3 \vee D_0^3) = \{1,2,3,4,5,6,7\} - [\{4,7\} \vee \emptyset]$, $N_0^3 = \{1,2,3,5,6\}$. Der Vergleich

$$\sum_{j \in N_0^3} a_{\overline{ij}} \leqq y_i^0 \ ?$$

zeigt, daß nicht alle Summen der negativen a_{ij}-Werte für $j \ N_0^3 < 0$ sind, denn $\sum_{j \in N_0^3} a_{\overline{4j}} = -3 - 3{,}5 < y_4^0 = -8{,}8$. Damit ist festgestellt, daß die für k = 0 versuchte Verbesserung der Lösung nicht möglich ist, weil auf diesem Weg keine zulässige Lösung erreicht wird. Es besteht die Anweisung, für diesen Fall alle v_j^0, deren Indizes zu der Menge N_0^3 gehören, zu streichen. Da auch keine weitere Lösung $k < 0$ $(J_k \subset J_3)$ mehr existiert, ist der Algorithmus beendet und die Lösung u^2 mit $J_2 = \{4,6\}$ und $z_2 = 8{,}9$ ist die optimale Lösung des Problems.

Die erste zulässige Lösung des Problems, welche mit Hilfe des Additiven Algorithmus ermittelt wurde, ist also gleichzeitig

die optimale, Diese Feststellung überrascht nicht, wenn man sich erinnert, daß die Auswahlkriterien für die Aufnahme der Variablen die Abstände zum zulässigen Bereich berücksichtigen. Mit Hilfe des Additiven Algorithmus gelangt man zwangsläufig zu Lösungen, die, sobald sie zulässig sind, in vielen Fällen gleichzeitig optimale Lösungen darstellen. Aufgrund dieser guten Konvergenz des Verfahrens war im vorliegenden Fall nur die Prüfung von 3 der $2^8 = 256$ Zuordnungen erforderlich.

Als allgemeingültiges und umfassendes Verfahren ist der Algorithmus von BALAS für die außerordentlich ganzzahlige Programmierung sicherlich genauso grundlegend wie das Simplex-Kriterium für die lineare Programmierung mit reellen Werten.

Für unser Beispiel erhalten wir: $x_4 = x_6 = 1$;
$x_1 = x_2 = x_3 = x_5 = x_7 = x_8 = 0$ und $z = 8{,}9$. Die Lösung des ursprünglichen Programms erhält man, indem für

$$x_j = \begin{cases} x'_j, & (j = 3,6) \\ 1 - x'_j, & (j = 1,2,4,5,7,8) \end{cases}$$

gesetzt wird mit: $x'_1 = x'_2 = x'_5 = x'_6 = x'_7 = x'_8 = 0$; $x'_3 = x'_4 = 0$.
Der maximale Gewinn beträgt $z' = 27$ DM . Eine Prüfung der Schlupfvariablen ergibt für das ursprüngliche Maximum-Problem:
$y_1 = -1$; $y_2 = -0{,}4$; $y_3 = 0$; $y_4 = 0{,}7$.

SCHLUSS

Der zunehmende Einsatz von Datenverarbeitungsanlagen, die Auseinandersetzung mit ihren logischen Möglichkeiten und die Tatsache, daß für eine Reihe wichtiger wirtschaftlicher Entscheidungsprobleme nur alternative Lösungen sinnvoll sind, haben zur Entwicklung eines neuen Teilgebietes der Unternehmensforschung geführt. Dieses bezeichnet man am zweckmäßigsten als zweiwertige Programmierung.

Am vielversprechendsten innerhalb dieses Problemkreises erscheinen Ansätze, welche auf die BOOLEsche Algebra zurückgehen, die ihrerseits auf der Aussagenlogik aufbaut: In diesen Ansätzen ist die Zweiwertigkeit der wirtschaftlichen Realität implizit berücksichtigt. Deshalb scheinen sie zur Konstruktion isomorpher Modelle betrieblicher Entscheidungssituationen besonders geeignet. Die Verwendung der herkömmlichen Verfahren der linearen Programmierung muß in diesem Zusammenhang als unzureichender Behelf angesehen werden.

Die Darstellung aller in diese Richtung gehenden Bemühungen und ihre Übertragung auf wirtschaftliche Fragestellungen war die Aufgabe der vorliegenden Arbeit. Daß diese Entwicklung noch nicht abgeschlossen ist, zeigen die jüngsten Veröffentlichungen auf diesem Gebiet[1].

---

[1] Ivanescu, P.L.; Rudeanu, S.: Pseudo-Boolean Methods for Bivalent Programming. Lecture of the First European Meeting of the Institute of Management Sciences and of the Econometric Institute, Warsaw, 2-7. Sept. 1966. Erschienen als Lecture Note in Mathematics Nr. 23 im Springer-Verlag, Berlin - Heidelberg - New York 1966. Hierin wird eine Reihe mathematischer Beiträge zur zweiwertigen Programmierung angekündigt.

ABKÜRZUNGEN

AMM	=	American Mathematical Monthly
BFuP	=	Betriebswirtschaftliche Forschung und Praxis
C.R.Acad.Sc.	=	Comptes Rendus de L'Academie des Sciences
JIE	=	The Journal of Industrial Engineering
JORSA	=	Operations Research The Journal of the Operations Research Society of America
MS	=	Management Science
NRLQ	=	Naval Research Logistics Quarterly
Ufo	=	Unternehmensforschung
ZfB	=	Zeitschrift für Betriebswirtschaft
ZfhF	=	Zeitschrift für handelswissenschaftliche Forschung
ZfbF	=	Zeitschrift für betriebswirtschaftliche Forschung
ZfN	=	Zeitschrift für Nationalökonomie
ZfgesSt	=	Zeitschrift für die gesamte Staatswissenschaft

LITERATURVERZEICHNIS

Ackermann, S.S.: Symbolic Logic: A Summary of the Subject and its Application to Industrial Engineering, in: JIE, Vol. VIII 1957 No. 5 S. 293 ff.

Adam, A.: Aussagenlogische Programmierung, in: Ufo 3. Bd. 1959, S. 136 ff.

Adam, A.: Messen und Regeln in der Betriebswirtschaft, Würzburg 1959

Adam, A.: Betriebskybernetik, in: Ufo 4. Bd. 1960, S. 44 ff.

Adam, A.: Systematische Datenverarbeitung bei der Auswertung von Versuchs- und Beobachtungsergebnissen, Würzburg 1963

Ahlsdorff, M.: Probleme des innerbetrieblichen Transportes in betriebstheoretischer Sicht, Diss. Göttingen 1965

Akers, S.B.; Friedman, J.: A non-numerical Approach to production scheduling Problems, in: JORSA, Vol. 3 1955 No. 4, S. 429 ff.

Akers, S.B.: A Graphical Approach to Production Scheduling Problems, in: JORSA Vol. 4 1956 No. 2, S. 244 ff.

Albach, H.: Maschinenbelegungspläne bei Einzelfertigung, Sonderdruck aus dem Jahrbuch 1965 des Landesamtes für Forschung des Landes Nordrhein-Westfalen, Köln und Opladen 1965

Algan, M.: Reihenfolgeprobleme und Graphentheorie, in: Ufo Bd. 8 Heft 2 1964, S. 53 ff.

Angermann, A.: Kybernetik und betriebliche Führungslehre, in: BFuP 11. Jg. 1959, S. 257 ff.

Angermann, A.:	Unternehmerische Entscheidungen und Operations Research, in: Betriebsführung und Operations Research, Hrsg. A. Angermann, Frankfurt/M. 1963, S. 9 ff.
Angermann, A.:	Entscheidungsmodelle, Frankfurt/M. 1963
Avondo-Bodino, G.:	Economic Applications of the Theory of Graphs, New York 1962
Balas, E.:	Un algorithme additif pour la resulution des programmes lineaires en variables bivalentes, in: C.R.Acad.Sc., Paris 258. Bd. 1964, S. 3817 ff.
Balas, E.:	Extension de l'algorithme additif a la programmation en nombres entiers et a la programmation non lineaire, in: C.R.Acad.Sc. 258. Bd. 1964, S. 5136 ff.
Balas, E.:	An Additive Algorithm for Solving Linear Programs with Zero-One Variables, in: JORSA, Vol. 13 1965 No. 4, S. 517 ff.
Balinski, M.L. and Quandt, R.E.:	On an Integer Program for a Delivery Problem, in: JORSA Vol. 12 1964 No. 2, S. 300 ff.
Balinski, M.L.:	Integer Programming: Methods, Uses, Computation, in: MS Vol. 12 1965 No. 3, S. 253 ff.
Beer, St.:	Kybernetik und Management, Frankfurt/M. 1963
Bellmann, R.:	Maximization over Discrete Sets, in: NRLQ Vol. 3 1956, S. 67 ff.
Benders, J.F.:	Partitioning Procedures for Solving Mixed Variables Programming Problems, in: Numerische Mathematik 1962, S. 238 ff.
Berg, R.:	Planung von Büroräumlichkeiten, in: Industrielle Organisation, Heft 9 1957, S. 337 ff.
Berge, C.:	The Theory of Graphs and its Applications, Londen - New York 1962

Bochenski, I.M.:	Formale Logik, 2. Aufl. Freiburg - München 1962
Bochenski, I.M.; Menne, A.:	Grundriß der Logistik, 3. Aufl. Paderborn 1965
Boole, G.:	The mathematical Analysis of Logic, Cambridge 1847, Neudruck 1947
Boole, G.:	An Investigation of the Laws of Thought, London 1854, Neudruck Chicago - London 1916
Camion, P.:	Une Methode de resolution par L'Algebre de Boole des problemes combinatoires ou interviennent des entiers, in: Cahiers Du Centre D'Etudes de Recherche Operationnelle, Vol. 2 1960 No. 3, S. 234 ff.
Carnap, R.:	Symbolische Logik, 2. Aufl. Wien 1960
Churchman, C.W.; Ackoff, R.L.; Arnoff, L.E.:	Operations Research. Eine Einführung in die Unternehmensforschung, Wien - München 1961
Cobham, A.; Fridshal, R.; North, J.H.:	An Application of Linear Programming to the Minimization of Boolean Functions, IBM-Research-Report RC-472 1961
Cobham, A.; North, J.A.:	Extensions of the Integer Programming Approach to the Minimization of Boolean Functions, IBM-Research-Report RC-915 1963
Courant, R. und Robbins, H.:	Was ist Mathematik? Berlin - Göttingen - Heidelberg 1962
Cushen, W.E.:	Symbolic Logic in Operations Research, in: Operations Research for Management, Hrsg. Closkey, J.F., Trefethen, F.N., Baltimore 1960, S. 187 ff.
Dantzig, G.B.; Fulkerson, D.R.; Johnson, S.:	Solution of a large Scale Traveling Salesman Problem, in: JORSA Vol. 2 1954 No 4, S.393 ff.
Dantzig, G.B.:	Discrete-variable Extremum Problems, in: JORSA Vol. 5 1957 No. 2, S. 266 ff.

Dantzig, G.B.:	On the Shortest Route through a Network, in: MS Vol. 6 1960 No. 2, S. 187 ff.
Dantzig, G.B.:	On the Significance of Solving Linear Programming Problems with some Integer Variables, in: Econometrica Vol. 28 1960 No.1, S. 30 ff.
Dantzig, G.B.:	Lineare Programmierung und Erweiterungen, Berlin - Heidelberg-New York 1966
Debry, R. et; Huyberechts, S.:	Quelques Aspects de la Theorie des Graphes, in: Cahiers Du Centre D'Etudes De Recherche Operationnelle, No. 2 1959, S. 29 ff.
Dinkelbach, W.:	Zum Problem der Produktionsplanung in Ein- und Mehrproduktunternehmen, Würzburg - Wien 1964
Dwinger, P.:	Introduction to Boolean Algebras, Würzburg 1961
Eaton, P.T.:	Logistik im Betrieb. Materialfluß gewinnt an Bedeutung, in: Der Volkswirt, 20. Jg. Nr. 24 1966, S. 1028 f.
Eccles, H.E.:	Logistics - What is it? in: NRLQ Vol. 1 1954, S. 5 ff.
Eichenberger, H.:	Ökonomische Methoden in der linearen Planungsrechnung, Hamburg - Berlin 1963
Erdmann, A.:	Das Gleichungssystem der doppelten Buchhaltung, in: ZfB 30. Jg. 1960, S. 231 ff.
Fischl, J.:	Logik. Ein Lehrbuch. Mit einem kurzen Abriß über Logistik, 2. Aufl. Graz - Wien - Altötting 1952
Flood, M.M.:	The Traveling Salesman Problem, in: JORSA Vol. 4 1956 No. 1, S. 61 ff.
Ford, L.R.; Fulkerson, D.R.:	Maximal Flow through a Network, in: Canadian Journal of Mathematics, Vol. VIII 1956, S. 399 ff.

Ford, L.R.; Fulkerson, D.R.: A Simple Algorithm for Finding Maximal Network Flows and an Application to the Hitchcock-Problem, in: Canadian Journal of Mathematics, Vol. IX 1957, S. 210 ff.

Ford, L.R.; Fulkerson, D.R.: A Suggested Computation für Maximal Multi-Commodity Network Flows, in: MS Vol. 5 1958 No. 1, S. 5 ff.

Ford, L.R.; Fulkerson, D.R.: Flows in Networks, Princeton 1965

Fortet, R.: L Algebre de Boole et ses Applications en Recherche Operationnelle, in: Cahiers Du Centre D'Etudes De Recherche Operationnelle 1959 No. 4, S. 5 ff.

Fortet, R.: Applications De L'Algebre de Boole en Recherche Operationnelle, in: Revue Francaise de Recherche Operationnelle, 4. Jg. 1960 No. 14, S. 17 ff.

Von Freytag-Löringhoff, B.: Logik, Ihr System und ihr Verhältnis zur Logistik, 3. Aufl. Stuttgart 1961

Fulkerson, D.R. and Dantzig, G.B.: Computations of Maximal Flows in Networks, in: NRLQ, Vol. 2 1955 No. 4, S. 277 ff.

Fulkerson, D.R.: Increasing the Capacity of a Network: The Parametric Budget-Problem, in: MS Vol. 5 1959 No. 4, S. 472 ff.

Fulkerson, D.R.: Flows in Networks, in: Recent Advances in Mathematical Programming, Hrsg. Graves R.L. and Wolfe, P., New York - Toronto - London 1963, S. 319 ff.

Gäfgen, G.: Theorie der wirtschaftlichen Entscheidung, Tübingen 1963

Geisler, M.A.: Logistics research and management science, in: MS Vol. 6 1960, S. 444 ff.

Geppert, F.:	Allgemeine Grundsätze zur Programmierung von Digital-Rechenautomaten, in: Ufo 3. Bd. 1959, S. 1 ff.
Giffler, B.; Thompson, G.L.:	Algorithmus for Solving Production Scheduling Problems, in JORSA, Vol. 8 1960 No. 4, S. 487 ff.
Giffler, B.:	Scheduling General Production Systems using Scheduling Algebra, in: NRLQ Vol. 10 1963 No. 3, S. 237 ff.
Glover, F.; Zionts, S.:	A Note on the Additive Algorithm of Balas, in: JORSA Vol. 13 1965 No. 4, S. 546 ff.
Gomory, R.E.:	Outline of an Algorithm for Integer Solutions to Linear Programs, in: Bulletin of the American Mathematical Society, Vol. 64 1958 No. 5, S. 275 ff.
Grochla, E.:	Zur Organisation des betrieblichen Planungsablaufs, in: ZfB 32. Jg. 1962, S. 702 ff.
Gutenberg, E.:	Grundlagen der Betriebswirtschaftslehre, 1. Bd. Die Produktion, 6. Aufl. Berlin - Göttingen - Heidelberg 1962.
Gutenberg, E.:	Grundlagen der Betriebswirtschaftslehre, 2. Bd. Der Absatz, 5. Aufl. Berlin - Göttingen - Heidelberg 1962
Hadley, G.:	Linear Programming, Reading - Palo Alto - London 1963
Hakimi, S.L.:	Optimum Distribution of Switching Centers in a Communication Network and some related Graph theoretic problems, in: JORSA, Vol. 13 1965 No. 3, S. 467 ff.
Harary, F.:	Graph Theoretic Methods in the Management Sciences, in: MS Vol. 5 1959, No. 4, S. 387 ff.
Hilbert, D. und Ackermann, W.:	Grundzüge der theoretischen Logik, 4. Aufl. Berlin - Göttingen - Heidelberg 1959

Hoss, K.:	Fertigungsablaufplanung mittels operationsanalytischer Methoden, Würzburg - Wien 1965
Ivanescu, P.L.; Rosenberg, I.; Rudeanu, S.:	Asupra determinarii minimelor functiilor Pseudobooleenne, in: Studii si cercetari matematice, 14. Jg. 1963, S. 359 ff.
Ivanescu, P.L.:	Programmation polynomiale en nombre entiers, in: Comptes Rendus (Hebdomadaires Des Seances) de l Academie des Sciences, Jg. 1964, Gr.1, S. 424 ff.
Ivanescu, P.L. and Rudeanu, S.:	Application of Pseudo-Boolean Programming to the Theory of Graphs, in: Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete Bd. 3 1964, S. 163 ff.
Ivanescu, P.L.:	Systems of Pseudo-Boolean Equations and Inequalities, in: Bulletin De L Academie Polonaise des Sciences, Vol. XII 1964, S. 673 ff.
Ivanescu, P.L.:	Systems of Pseudo-Boolean Equations and Inegualities,in: Bulletin De L'Academie Polonaise des Sciences, Vol. XII 1964, S. 673 ff.
Ivanescu, P.L.:	Pseudo Boolean Programming and Applications, Lecture Note in Mathematics No. 9, Berlin - Göttingen - Heidelberg 1965
Ivanescu, P.L.; Rudeanu, S.:	A Pseudo-Boolean Approach to Matching Problems in Graphs with Applications to Assigment and Transportation Problems, unveröffentlichtes Manuskript, Bukarest 1966
Ivanescu, P.L.; Rudeanu, S.:	Pseudo-Boolean Methods for Bivalent Programming, Lectur Note in Mathematics No. 23, Berlin - Heidelberg - New York 1966
Kade, G.:	Die logischen Grundlagen der mathematischen Wirtschaftstheorie als Methodenproblem der theoretischen Ökonomie, Diss. Berlin 1958

Kemeny, J.G.; Snell, J.L.; Thompson, G.L.:	Einführung in die endliche Mathematik, Ludwigshafen 1963
Kern, W.:	Gestaltungsmöglichkeit und Anwendungsbereich betriebswirtschaftlicher Planungsmodelle, in: ZfhF 14. Jg. 1962, S. 167 ff.
Klaus, G.:	Moderne Logik. Abriß der formalen Logik, Berlin 1965
König, D.:	Theorie der endlichen und unendlichen Graphen, New York 1950
Kosiol. E.:	Modellanalyse als Grundlage unternehmerischer Entscheidungen, in: ZfhF 13. Jg. 1961, S. 318 ff.
Kosiol, E.:	Betriebswirtschaftslehre und Unternehmensforschung, in: ZfB 34. Jg. 1964, S. 743 ff.
Kosiol, E. mit Szyperski, N. und Chmielewicz, K.:	Zum Standort der Systemforschung im Rahmen der Wissenschaften, in: ZfbF 17. Jg. 1965, S. 337 ff.
Koreimann, D.:	Kybernetische Grundlagen der Betriebswirtschaft, in: BFuP 17. Jg. 1965, S. 617 ff.
Kramer, R.:	Die betriebswirtschaftliche Bedeutung von Information und Kommunikation, insbesondere für die Struktur des Betriebes, Diss. Mannheim 1962. Veröffentlicht auch unter dem Titel: Information und Kommunikation, Berlin 1965
Kreko, B.:	Lehrbuch der linearen Programmierung, Berlin 1965
Krelle, W.:	Ganzzahlige Programmierungen. Theorie und Anwendungen in der Praxis, in: Ufo 2. Bd. 1958, S. 161 ff.
Krelle, W.:	Gelöste und ungelöste Probleme der Unternehmensforschung, in: Sitzungsbericht vom 4.10.1961 der Arbeitsgemeinschaft für Forschung des Landes Nordrhein-Westfalen, Köln und Opladen 1961

Kulhavy, E.:	Operations Research, Wiesbaden 1963
Ledley, R.S.:	Mathematical Foundations and Computational Methods for a Digital Logic Machine, in: JORSA Vol. 2 1954, S. 24 ff.
Lindsay, F.A.:	New Techniques for Management Decision Making, New York - London - Toronto 1963
Löwenheim, L.:	Über die Auflösung von Gleichungen im logischen Gebietekalkül, in: Mathematische Annalen 68. Bd. 1910, S. 169 ff.
Lorenzen, P.:	Formale Logik, 2. Aufl. Berlin 1962
Lücke, W.:	Die Liquidität im Entscheidungsmodell, in: Gegenwartsfragen der Unternehmensführung. Festschrift zum 65. Geburtstag von W. Hasenack, Herne - Berlin 1966, S. 323 ff.
Maghout, K.:	Applications de l'Algebre de Boole a la Theorie des Graphes et aux programmes lineaires et quadratiques, in: Cahiers Du Centre D Etudes De Recherche Operationnelle, Vol. 5 1963, S. 21 ff.
Manne, A.S.:	Plant Location under Economics of Scale Dezentralization and Computation, in: MS Vol. 11 1964, S. 213 ff.
Markowitz, H.M.; and Manne, A.S.:	On the Solution of Discrete Programming Problems, in: Econometrica, Vol. 25 1957, S. 84 ff.
Mayhugh, J.O.:	On the mathematical theory of schedules, in: MS Vol. 11 1964 No. 2, S. 289 ff.
Menne, A.:	Was ist und was kann Logistik? Paderborn 1957
Mertens, P.:	Betriebliche Dokumentation und Information, Meisenheim am Glan 1965

Michejew, W.M.:	Über Mengen, die eine größte Anzahl paarweise nicht vergleichbarer Boolscher Vektoren enthalten, in: Probleme der Kybernetik, Bd. 2, Berlin 1963, S. 75 ff.
Mirasol, N.M.:	A Quening Approach to Logistics Systems, in: JORSA Vol. 12 No. 5 1964, S. 707 ff.
Morgenstern, O.:	Logistik und Sozialwissenschaft, in: ZfN Bd. VII 1936, S. 1 ff. Wiederabdruck in: Logik der Sozialwissenschaften. Hrsg. Topitsch, E., Köln - Berlin 1965, S. 181 ff.
Morgenstern, O.:	Note on the Formulation of the Theory of Logistics, in: NRLQ Vol. 2 1955, S. 129 ff.
Neidhardt, P.:	Informationstheorie und automatische Informationsverarbeitung, 2. Aufl. Stuttgart 1964
Nelson, R.I.:	Simplest Normal Truth Functions, in: Journal of Symbolic Logic, Vol. 26 1954, S. 105 ff.
Neumann, J. v. und Morgenstern, O.:	Spieltheorie und wirtschaftliches Verhalten, Würzburg 1961
Nürck, R.:	Unternehmensführung - Ein Regelungsproblem, in: BFuP 12. Jg. 1960, S. 230 ff.
Nürck, R.:	Steuerung, Regelung, Kybernetik, Wirtschaftskybernetik, in: BFuP 17. Jg. 1965, S. 329 ff.
Nürck, R.:	Wirtschaftskybernetik. Ein Schlüssel zur Ganzheitsbetrachtung, in: ZfB 35 Jg. 1965, S. 573 ff.
Ore, O.:	Graphs and their Uses, New York 1963
Peirce, J.R.:	Phänomene der Kommunikation, Düsseldorf - Wien 1965
Pfiffner, J.M.; Sherwood, F.P.:	Administrative Organization, New York 1960
Pietzsch, J.:	Die Information in der Unternehmung, Köln und Opladen 1965

Quine, W.V.: A way to simplify Truth Functions. in: AMM Vol. 62 1955, S. 627 ff.

Quine, W.V.: The Problem of Simplifying Truth Functions, in: AMM Vol. 59 1962, S. 521 ff.

Roth, J.P.: Algebraic Topological Methods for the Synthesis of Switching Systems, I. Transactions of the American Mathematical Society, Vol. 88 1958, S. 301 ff.

Rechenberg, P.: Grundzüge digitaler Rechenautomaten, München und Wien 1964

Riester, W.F.: Planungsmodelle im Steinkohlenbergbau, in: Gegenwartsfragen der Unternehmensführung, Festschrift zum 65, Geburtstag von W. Hasenack, Herne - Berlin 1966, S. 225 ff.

Root, J.G.: An Application of Symbolic Logic to a Selection problem, in: JORSA, Vol. 12 1964 No. 4, S. 519 ff.

Rosenberg, I.; Ivanescu, P.L.; Rudeanu, S.: An Application of Discrete Linear Programming of Boolean Functions, in: Revue De Mathematiques Pures et Appliques Vol. III 1963 No. 3, S. 459 ff.

Rosenblatt, D.: On Some Aspects of Models of Complex Behavioral Systems, in: Machol, R.E.: Information and Decision Processes, New York - Toronto - London 1960, S. 62 ff.

Sauter, B.: Die Bedeutung der Kybernetik für die Betriebsorganisation, Diss. Mannheim 1959

Sikorski, R.: Boolean Algebras, 2. Aufl. Berlin - Göttingen - Heidelberg - New York 1964

Smidt, R.M.; Reis, I.L.: Symbolic Logic and Plant Location, in: JIE Vol. XIV No. 1 1963, S. 18 ff.

Sommer, F.: Mathematik für Studenten der Wirtschaftswissenschaften, Berlin - Göttingen - Heidelberg 1962

Speiser, A.:	Digitale Rechenanlagen, 2. Aufl. Berlin - Heidelberg - New York 1965
Szwarc, W.:	Solution of the Akers - Friedman Scheduling Problem, in: JORSA Vol. 8 1960 No. 6, S. 782 ff.
Szyperski, N.:	Zur Problematik der quantitativen Terminologie in der Betriebswirtschaftslehre, Berlin 1962
Schmidt, A.:	Mathematische Gesetze der Logik I. Berlin - Göttingen - Heidelberg 1960
Schweitzer, M.:	Probleme der Ablauforganisation in Unternehmungen, Berlin 1964
Stahlmann, J.:	Organisation, Entscheidung und Kommunikation, Diss. Göttingen 1960
Stranzky, R.:	Kybernetik ökonomischer Reproduktion, Berlin 1966
Tempel, K.-H.:	Wartezeitmodelle, Diss. Göttingen 1964
Vaszonyi, A.:	Die Planungsrechnung in Wirtschaft und Industrie, Wien - München 1962
Van der Waerden, B.L.:	Algebra, 1. Teil, Berlin - Göttingen - Heidelberg 1960
Wenke, K.:	Matrizenmodelle in der Großindustrie, in: Anwendungen der Matrizenrechnung auf wirtschaftliche und statistische Probleme, Hrsg. Adam, A.; Ferschl, F. u.a., Würzburg 1963, S. 112 ff.
Weyh, U.:	Elemente der Schaltungsalgebra, 3. Aufl. München 1964
Whitesitt, I.E.:	Boolesche Algebra und ihre Anwendungen, Braunschweig 1964
Wiener, N.:	Kybernetik. 2. Aufl. Düsseldorf - Wien 1963
Wittmann, W.:	Unternehmung und unvollkommene Information, Köln und Opladen 1959

Zemanek, H.: Logische Algebra und Theorie der Schaltnetzwerke, in: Taschenbuch der Nachrichtenverarbeitung, Hrsg. K. Steinbuch, Berlin - Göttingen - Heidelberg 1962, S. 100 ff.

Zimmermann, W.: Modellanalytische Verfahren zur Bestimmung optimaler Fertigungsprogramme, Berlin 1966

LEBENSLAUF

Am 29. März 1938 wurde ich, Gösta-Bernd Friedrich Martin Ihde, als Sohn des selbständigen Automobilhändlers Urban Ihde und seiner Ehefrau Jutta-Gisela, geb. Steuer, in Rostock/Mecklenburg geboren.

Infolge der Kriegswirren wurde ich erst im Herbst 1946 eingeschult und besuchte in den folgenden sieben Jahren verschiedene Volksschulen, da ich als Kind "kapitalistischer" Eltern zur Oberschule nicht zugelassen wurde.

Im Februar 1953 flohen meine Eltern in die Bundesrepublik und siedelten sich schließlich - nach Durchlaufen mehrerer Flüchtlingsläger - in Goslar/Harz an. Hier wurde ich in die Obertertia des Ratsgymnasiums aufgenommen.

Im Frühjahr 1959 wurde ich unmittelbar nach meiner Reifeprüfung zum Wehrdienst einberufen.

Das wirtschaftswissenschaftliche Studium habe ich im Sommersemester 1960 in Marburg aufgenommen, 1961 in Saarbrücken fortgesetzt und im Frühjahr 1964 in Göttingen mit der Prüfung für Diplom-Kaufleute abgeschlossen.

Nach einer mehrmonatigen Militärübung wurde ich vom 1. Dezember 1964 an zunächst als wissenschaftlicher Angestellter am Betriebswirtschaftlichen Seminar der Georg-August-Universität beschäftigt. Seit Juli 1965 verwalte ich bei Herrn Professor Dr. W. Lücke, dem Direktor des Seminars für Betriebswirtschaftliche Produktionsforschung, die Planstelle eines wissenschaftlichen Assistenten.

BAMBERGER FOTODRUCK - Rudolf Rodenbusch